Neuron-Synapse Level problem decomposition method for Cooperative Coevolution of Recurrent Networks for Time Series Prediction

Ravneil Nand
School of Computing Information and Mathematical Sciences
University of South Pacific, Suva, Fiji.
Artificial Intelligence and Cybernetics Research Group, Software Foundation, Nausori, Fiji.
Email: ravneiln@yahoo.com

***Abstract*—The decomposition of a particular problem can become a tiresome task if little connections between the elements is needed. In cooperative coevolution of recurrent networks, synapse and neuron level are the two noteworthy problem decomposition methods. Through combination of both of the problem decomposition methods, the individual problem decomposition methods can share its strengths to solve the problem at hand better. In this paper, a recently proposed problem decomposition method known as Neuron-Synapse problem decomposition method is modified for Elman recurrent neural networks. The results reveal that the proposed method has got better results in selected datasets when compared to standalone methods. The results are better in some cases for proposed method when compared to other approaches from the literature.**

I. Introduction

Neural networks (NN) are computational intelligence tactic spawned from biological neural systems (the brain) [1]. They consist of a group of interconnected neurons known as layers which exchange messages between different neurons. The links between different layers are called weights (synapses). Through the use of datasets these synapses can be modified based on training algorithm. The neural network is applied for pattern classification, time series forecasting, function approximation, clustering, function optimization and robot control [2].

The neural network is mainly categorized into feedforward and recurrent network architectures [3]. Feedforward neural network architectures occasionally have three layers, which are input, hidden and output layers while the recurrent network has an additional context layer. Figure 1 shows a basic structures of a feedforward neural network and an Elman recurrent neural network.

Neuro-evolution is the use of evolutionary algorithms for training artificial neural networks. Neural networks have the ability to solve complex functions as well as model open dynamical framework [4]. Through the use of evolutionary algorithms (EAs) as a part of neural network training and design, it has been possible to design a better algorithm when compared with traditional gradient descent based methods [5].

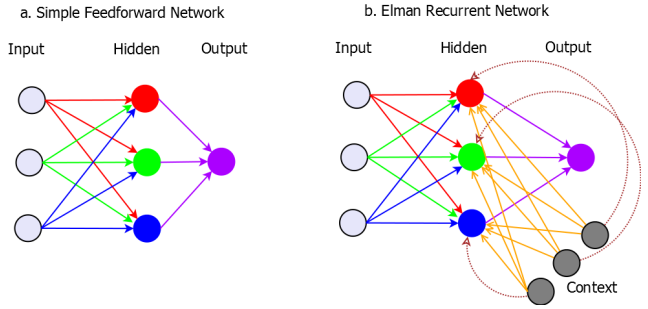


Fig. 1. An outline of the basic structures of (a) feedforward and (b) recurrent neural networks.

Time series prediction incorporates time series data at various intervals to forecast future values [6] [7]. Being chaotic or noisy in nature are two qualities of time series dataset, which are highly sensitive to initial conditions and noise [7]. At any given point in time for such datasets, slight changes in the data values can bring about entirely different output. There have been applications ranging from climate prediction [8] to financial prediction [9] for the time series prediction.

Evolutionary computation techniques, such as cooperative coevolution, are used to train and test artificial neural networks [10], [11], [12], [13]. Cooperative coevolution (CC) is a biologically inspired computation tactic that breaks a problem down into smaller pieces known as subcomponents [10].

The subcomponents of CC are implemented as sub-population. The sub-populations cooperatively coevolve with a specific end result to achieve an ideal solution. The sub-populations of the CC are evolved in isolation where the cooperation is achieved during fitness evaluation of each sub-population. In Figure 2, a complete solution is shown that combine the best individuals from all the sub-populations. Problem decomposition is a major challenge in cooperative neuro-evolution.

To address problem decomposition [12], [14] challenges, there is some work done in the area of CC that include adaptation of problem decomposition [15] and the application

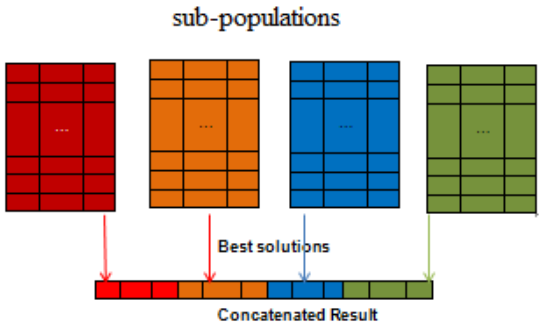


Fig. 2. Cooperative Coevolution framework that forms complete solution from different sub-population.

of problem decomposition methods for time series prediction [13], [16]. CC has showed some good results in comparison to other or similar methods in the literature [13] in terms of problem decomposition.

Cooperative Synapse Neuro-evolution (CoSyNE) [12] is one of the problem decomposition methods that has been introduced to train the feedforward and recurrent networks on pole balancing problem. The architecture of CoSyNE has a single interconnection which is either the weight or bias in the network which act as a subcomponent. Another problem decomposition called neuron-based sub-population (NSP) was introduced later [17]. The performance was better for NSP than CoSyNE for pattern classification applications [17]. In NSP, the neurons within the network act as the reference point such as hidden neuron.

The use of both NSP and CoSyNE problem decomposition method in a competitive island cooperative coevolution method (CICC) showed good performance [18]. In CICC, different problem decomposition methods acts as islands that goes through competition with other island in training and testing of neural network. The island with best individuals at each round transfers its solution to the another island through collaboration. CICC extracted different problem decomposition features to solve a problem in hand better. Instead of using different problem decomposition as islands the synapse level can be used in the region where more diversity in search is needed and neuron level can be applied where more decision making is required [19].

In CC [10], a better problem decomposition tactic relies on the type of the optimization problem available [14]. One challenge of CC includes on finding a best problem decomposition method that can identify inter-dependencies between variables [14]. The problem in the traditional CC algorithm gets divided in a single sub-population [10], which has been effective for fully separable problems.

As for in this paper, the Neuron-Synapse Level (NSL) problem decomposition [19], is being modified to suit Elman recurrent network. The main goal to modify the algorithm for recurrent neural network was to see the potential of the new problem decomposition in different network. As for NSL problem decomposition, it creates fewer subpopulations than SL method and a higher number than NL method.

The rest of the paper is organized as follows. In Section 2, the proposed problem decomposition is discussed in detail. The Section 3 shows the experiments and results with discussion and Section 4 conclude the paper with a discussion of future extensions of the paper.

II. Proposed Method

In cooperative coevolution, Synapse level (SL) problem decomposition method provides the most diversity while Neuron level (NL) problem decomposition method provides less diversity and more prominence is given on the interacting variables (weights) [5]. The challenge in cooperative coevolution is to have a balance between diversity and interacting variables. Therefore, combining of the two encoding schemes together can solve the problem [19].

This hybrid problem decomposition method is called Neuron-Synapse level (NSL), which has been already tested on feed forward networks [19]. NSL decomposes the problem into a lower level as done in NL problem decomposition method. As for NSP, each subcomponent consists of incoming connections associated with neurons in the hidden and output layers. The calculation of the actual output is given by the total of all the outputs generated at each neuron.

As for the application of NSL to recurrent network, one main difference appears is the use of context layer. In the modified version of NSL for recurrent network, weight links from context layer is also included for decomposition. NSL allows to search more efficiently than other methods such as NL. The main difference is at hidden-output layer in which the decomposition is at Synapse Level rather than a number of output neurons.

NSL employs a single subcomponent for each neuron that groups synapses that are connected from the hidden neuron. The break down of the subcomponents of NSL is shown in Fig.3 and composed as follows:

- Input to hidden layer subcomponents: comprises of all the weight connections between all input neuron to one specific hidden neuron and the bias. Input-hidden layer weights are decomposed as neuron level [17]:

- Context layer subcomponents: comprises of all the weight connections between each of the neuron in the context layer to all the hidden layer neurons.

- Hidden to output layer subcomponents: comprises of all weight connections from one specific neuron in the hidden layer to all output layer neurons and the bias.

The total number of subcomponents in NSL is equal to the total number of hidden neurons plus the number of weight connections between the hidden and context layer and plus the number of weights and biases within hidden layer and output layer in the neural network. The subcomponents are implemented as subpopulations.

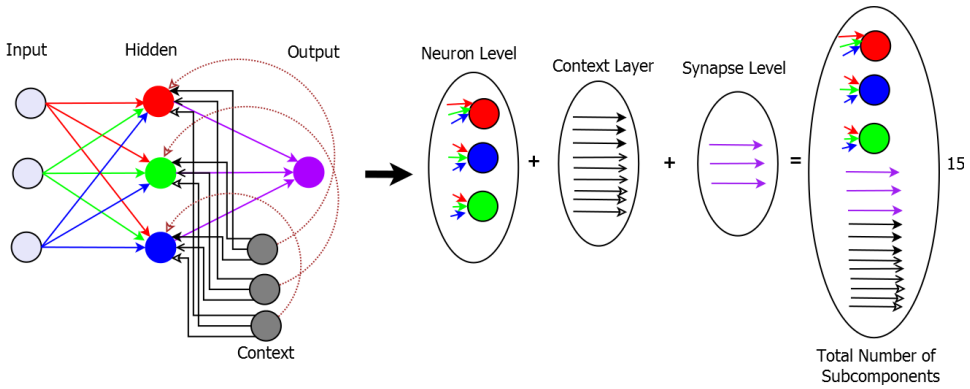


Fig. 3. Neuron-Synapse Level Problem Decomposition (NSL) breaks down the neural network into specific regions and encodes it into the respective sub-populations.

Algorithm 1: NSL for Training Recurrent Networks

Step 1: Decompose the problem into subcomponents according to NSL. Here problem is decomposed into input-hidden, context, and hidden-output layer subcomponents.
Step 2: Encode of subpopulation. The encoding of subpopulations are done with reference to hidden layer.
Step 3: Initialize and cooperatively evaluate each subpopulation in Step 2.
foreach *Cycle until termination* **do**
  foreach *Sub-population* **do**
    foreach *Depth of n Generations* **do**
      Select and create new individuals using genetic operators
      Cooperative evaluate the new individuals
      Add new individuals to the subpopulation
    end
  end
end

The NSL method is used in training recurrent network and the steps taken during training is shown in Algorithm 1. In Step 1 of the algorithm, the problem is decomposed in the number of subcomponents according to proposed NSL method. As for Step 2, the problem is encoded based on the number of neurons in the hidden layer and context layer.

The algorithm proceeds to Step 3 once the encoding of the network is done. In the next step, real random values are utilized to initialize all the individuals within the population. The fitness of all the individuals are then evaluated based on selection of arbitrary individuals present in the rest of the sub-populations.

The fitness is calculated based on concatenated individuals from all of the sub-populations. These individuals are later evaluated with the training data set once encoded into a neural network. Evaluation of the fitness of each member of the subpopulation is done through cooperation from the other subsets [10].

In this step, the evolution also takes place using genetic operators. For a fixed number of generations each subpopulation gets evolved where a cycle is completed when all the subpopulations have been successfully evolved within the subcomponent.

Here the subpopulation gets evolved using the genetic algorithm. The genetic algorithm used is called generalized generation gap parent centric crossover (G3-PCX) [20] where one parent selected is having the best fitness while rest are selected randomly from the subpopulation to generate the new individuals.

Once subpopulation gets evolved, the fitness of the new individuals is evaluated. Here two new parents get randomly selected from the main subpopulation for comparison and later can get changed by the new individuals.

The evaluation for any member of the subpopulation is done by joining the best members from the rest of the sub-populations. These subpopulations gets evolved for a allocated number of epoch or generations. As maximum fitness evaluations of 50,000 are reached for the network, the generalization performance is tested.

The synapse and neuron level problem decomposition methods create sub-population in relation to hidden neurons which affected diversity [19]. As for the proposed method, the number of sub-populations is also based on hidden neurons, hence, more diversity is given due to increasing in the sub-population while the network problem is approached as both separable (synapse level) and non-separable (neuron level) training problem [14].

III. Experiments and Results

This section shows the experimental setup and results based on cooperative coevolution on chaotic (noisy) time series problems to train recurrent networks. The modified hybrid method NSL and the two established problem decomposition methods are compared at first and later NSL is compared with results from literature.

1) Experimental Setup: The dataset is reconstructed using Taken's embedding theorem [21] before it was used. Two important conditions of *time delay (T)* and *embedding dimension (D)* [21] is taken into account while doing reconstruction. To reproduce important features of the unreconstructed dataset, the values *T* and *D* needs special consideration in terms of vector [22].

The hybrid model is being tested on five different datasets. The five datasets that are tested are Mackey-Glass [23], Lorenz [6], Sunspot [24], ACI Worldwide Inc. [25] and Seagate [25] datasets.

Firstly, to generate the *Mackey-Glass time series* the differential equation 1 is used.

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

The size of dataset used in the experiment for Mackey-Glass time series is 1000 where the first 500 values are used for training the neural network and the remaining 500 are used to test the neural network. The embedding dimension $D = 3$ and $T = 2$ is used during reconstruction. The time series is scaled in the range [0,1].

Secondly, *Lorenz time series* is chaotic in nature and is simulated time series as Mackey-Glass. The size of dataset used in the experiment for Lorenz time series is 1000 where the first 500 values are used for training the neural network and the remaining 500 are used to test the neural network. The embedding dimension $D = 3$ and $T = 2$ is used during reconstruction. Here the time series is scaled in the range [-1,1].

Thirdly, *Sunspot time series* is a real world problem which gives an indication of the solar activities for solar cycles which impact Earth's climate change [24]. The size of dataset used in the experiment for Sunspot time series is 2000 where the first 1000 values are used for training the neural network and the remaining 1000 are used to test the neural network. The embedding dimension $D = 5$ and $T = 2$ is used during reconstruction. Here the time series is scaled in the range [-1,1].

Fourthly, *ACI Worldwide Inc. time series* is a financial dataset taken from the NASDAQ stock exchange [25]. The size of dataset used in the experiment for Lorenz time series is 800 where the first 400 values are used for training the neural network and the remaining 400 are used to test the neural network. The embedding dimension $D = 5$ and $T = 2$ is used during reconstruction. Here the time series is scaled in the range [0,1].

Lastly, Seagate Technology PLC is also a financial time series dataset having daily closing stock prices from December 2006 to February 2010 [25]. The size of dataset used in the experiment for Seagate time series is 800 where the first 400 values are used for training the neural network and the remaining 400 are used to test the neural network. The embedding dimension $D = 5$ and $T = 2$ is used during reconstruction. Here the time series is scaled in the range [0,1].

The scaling of the five time series dataset is done in the range of [0,1] and [-1,1] as in the literature to provide fair comparison.

The experiment is run 50 times for each dataset and to end each run, the maximum number of function evaluations of 50 000 needs to be reached.

Sigmoid units are employed for the Mackey-Glass, Seagate, and ACI Worldwide Inc. Hyperbolic tangent unit is used for Lorenz and Sunspot time series. Root mean squared error (RMSE) is used to evaluate the performance of the hybrid model, as was used in [13]. A pool size of 2 parents and 2 individuals are put in the G3-PCX algorithm based on *generation gap model* [20] for selection as done in the literature [13].

The reason to use G3-PCX algorithm to evolve all the sup-populations in the hybrid method was due to its good performance with cooperative coevolution as mentioned in [5]. The population size used in our entire experiments is 300 since it has gave the best results during trial experiment.

To popular problem decomposition methods (Synapse level and Neuron level) for comparision are used for decomposing the problem into subcomponents. As the maximum number of function evaluations which is 50,000 are reached by the respective cooperative coevolution methods (NSL, NL, and SL) the training experiment is terminated.

The prediction performance of the proposed method is measured by the Root Mean Squared Error (RMSE) and Normalized Mean Squared Error (NMSE) as done in similar algorithms [18], [13]. RMSE is also used as fitness function during evolution. The Equation 2 and Equation 3 are used to measure the performance of the training algorithm.

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

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

where y_i, is observed data, $\hat{y}_i$ is predicted data and $\bar{y}_i$ is average of observed data. And N is the observed data's length. In order to compare the results with the literature, these two performance measures are used.

A. Results

This subsection reports on the performance of proposed model based on the setup given previously.

In Tables I - V, the results of hybrid model is compared with the results of standalone cooperative coevolution methods on different number of hidden neurons. The results are based on 95 percent confidence interval on RMSE and shows the best run results based on different methods on different number of hidden neurons .

The column heading *Training* in the given tables shows the train average with train error sum and the heading *Generalisation* shows the test average with test error sum while the

TABLE I
THE RMSE BASED RESULTS OF NL, SL AND NSL FOR THE MACKEY-GLASS TIME SERIES

Method	H	Training	Generalisation	Best
RNN-NL	3	**0.0113 ± 0.0010**	**0.0114 ± 0.0011**	**0.006**
	5	0.016 ± 0.0011	0.0437 ± 0.0011	0.012
	7	0.0168 ± 0.0012	0.0434 ± 0.011	0.015
RNN-SL	3	0.0181 ± 0.0021	0.0182 ± 0.0021	0.010
	5	**0.0164 ± 0.0019**	**0.0164 ± 0.0019**	**0.008**
	7	0.0191 ± 0.0041	0.0191 ± 0.0041	0.009
RNN-NSL	3	0.0137 ± 0.0021	0.0140 ± 0.0021	0.005
	5	0.0088 ± 0.0018	0.0091 ± 0.0019	0.005
	7	**0.0052 ± 0.00052**	**0.0054 ± 0.00053**	**0.003**

heading *Best* shows the best test RMSE. The best results for each method are highlighted in bold.

In the Table I, the Mackey-Glass time series dataset results are shown. It was seen that NSL gave better generalization than the other two methods being compared to. The hybrid method showed better generalization performance and recorded best training value with seven hidden neurons.

Table II, illustrates the evaluation of the Lorenz time series problem. It was seen that the hybrid model under performed than the other two methods in terms of generalization. It was seen that the generalization performance and training of the NSL and the other two methods increases when the number of the hidden neuron increases. The best result for NSL came from seven hidden neurons.

TABLE II
THE RMSE BASED RESULTS OF NL, SL AND NSL FOR THE LORENZ TIME SERIES

Method	H	Training	Generalisation	Best
RNN-NL	3	0.0177 ± 0.0015	0.0184 ± 0.0016	0.007
	5	**0.0132 ± 0.0014**	**0.0136 ± 0.0015**	**0.003**
	7	0.0143 ± 0.0015	0.0147 ± 0.0016	0.005
RNN-SL	3	0.0209 ± 0.0024	**0.0214** ± 0.0025	0.008
	5	0.0168 ± **0.0020**	0.0175 ± **0.0021**	**0.004**
	7	**0.0164** ± 0.0028	**0.0172** ± 0.0030	0.008
RNN-NSL	3	0.0348 ± 0.0169	0.0358 ± 0.0168	0.010
	5	0.0200 ± **0.0029**	**0.0202 ± 0.0030**	0.009
	7	**0.0199** ± 0.0031	0.0213 ± 0.0034	**0.008**

Table III the Sunspot time series dataset results are shown. In Sunspot time series, noise is present since it is real-world dataset. The hybrid method NSL was unable to outperform both of the methods it is being compared to in terms of training and testing. The best result for NSL was given by five hidden neurons whereas for the other two methods it was seven hidden neurons.

In Table IV, the ACI Worldwide Inc. time series problem results are illustrated. The time series is real time series where noise is present as in Sunspot time series since it is also real-world dataset. As for this time series, the NSL performed better than the other two methods its being compared to in terms of generalization and best results. For NSL, seven hidden neurons have given best results.

TABLE III
THE RMSE BASED RESULTS OF NL, SL AND NSL FOR THE SUNSPOT TIME SERIES

Method	H	Training	Generalisation	Best
RNN-NL	3	0.0207 ± 0.0022	**0.0512 ± 0.0123**	0.017
	5	0.0179 ± 0.0019	0.0537 ± 0.0128	0.017
	7	**0.0165 ± 0.001**	0.0566 ± 0.0155	**0.015**
RNN-SL	3	0.0207 ± 0.0022	**0.0512 ± 0.0123**	0.017
	5	0.0179 ± 0.0019	0.0537 ± 0.0128	0.017
	7	**0.0165 ± 0.001**	0.0566 ± 0.0155	**0.015**
RNN-NSL	3	0.0284 ± 0.0044	**0.0612 ± 0.0153**	**0.016**
	5	**0.0250 ± 0.0027**	0.0703 ± 0.0221	0.020
	7	0.0273 ± 0.0112	0.1334 ± 0.0400	0.025

TABLE IV
THE RMSE BASED RESULTS OF NL, SL AND NSL FOR THE ACI WORLDWIDE INC. TIME SERIES

Method	H	Training	Generalisation	Best
RNN-NL	3	0.0207 ± 0.0004	0.0212 ± 0.0013	0.019
	5	0.0203 ± 0.0003	0.0204 ± **0.0002**	0.019
	7	**0.0201 ± 0.0002**	**0.0204** ± 0.0004	**0.019**
RNN-SL	3	0.0226 ± 0.0007	0.0219 ± 0.0008	0.019
	5	**0.0220** ± 0.0007	**0.0211 ± 0.0005**	**0.019**
	7	0.0211 ± **0.0005**	0.0211 ± 0.0006	0.019
RNN-NSL	3	0.0247 ± 0.0010	0.0198 ± 0.0013	0.015
	5	0.0238 ± 0.0018	0.0187 ± 0.0017	0.015
	7	**0.0225 ± 0.0007**	**0.0173 ± 0.0008**	**0.015**

TABLE V
THE RMSE BASED RESULTS OF NL, SL AND NSL FOR THE SEAGATE TIME SERIES

Method	H	Training	Generalisation	Best
RNN-NL	3	0.0351 ± 0.0549	0.3220 ± 0.065	0.032
	5	0.0351 ± 0.04482	0.3223 ± 0.0742	0.032
	7	**0.0351 ± 0.0226**	**0.3215 ± 0.05511**	**0.032**
RNN-SL	3	0.0351 ± 0.06283	0.3216 ± 0.06534	**0.031**
	5	0.0351 ± 0.05319	**0.3215 ± 0.0562**	0.032
	7	**0.0350 ± 0.0480**	0.3217 ± 0.0654	0.032
RNN-NSL	3	0.0209 ± 0.0009	**0.1850 ± 0.0359**	**0.028**
	5	**0.0193 ± 0.0004**	0.2293 ± 0.0519	0.055
	7	0.0195 ± 0.0004	0.1987 ± 0.0470	0.034

In Table V, the Seagate time series problem is evaluated. For this time series, the NSL method outperformed the other two methods, NL, and SL. Three hidden neurons have given the best result for NSL.

The best results of NSL from Tables I - V with NMSE are given in Tables VI - X for comparison with some of the related results of methods in literature. The hybrid NSL method has shown good performance in some of the datasets when vied to other methods in the literature.

In Figures 4 - 6, the predicted test results of NSL method with original results on different datasets is shown. The error graph is showing different between original and predicted results.

In Table VI, the best result on Mackey-Glass time series

TABLE VI
MACKEY-GLASS TIME SERIES RESULTS OF NSL BEING COMPARED WITH LITERATURE

Prediction Method	RMSE	NMSE
AMCC-RNN [16]	7.53E-03	3.90E-04
Locally linear neuro-fuzzy model (2006) [26]	9.61E-04	
SL-CCRNN [13]	6.33E-03	2.79E-04
NL-CCRNN [13]	8.28E-03	4.77E-04
CICC-RNN [18]	3.99E-03	1.11E-04
FNN-NSL [19]	4.86E-03	4.48E-05
Proposed RNN-NSL	2.99E-03	1.93E-04

TABLE VII
LORENZ TIME SERIES RESULTS OF NSL BEING COMPARED WITH LITERATURE

Prediction Method	RMSE	NMSE
RBF with orthogonal least squares (2006) [26]		1.41E-09
Locally linear neuro-fuzzy model (2006) [26]		9.80E-10
SL-CCRNN [13]	6.36E-03	7.72E-04
NL-CCRNN [13]	8.20E-03	1.28E-03
CICC-RNN [18]	3.55E-03	2.41E-04
FNN-NSL [19]	2.34E-03	2.87E-05
Proposed RNN-NSL	8.16E-03	1.28E-03

problem is being compared to works in literature. The proposed method was able to outperform all the methods it is being compared to in terms of RMSE. The competitive island model used in CICC-RNN was not beaten by the proposed method in terms of NSME.

The best result of Lorenz time series problem is compared to works in literature in Table VII. The proposed method has only been able to outperform NL-CCRNN method. Even it was not close to the results of FNN-NSL.

In Table VIII, the best result of the Sunspot time series problem is compared with results in the literature. The NSL method has shown to outperform most of the methods it is compared to expect for CICC-RNN and FNN-NSL methods. The method has shown competitive results.

The best result of the ACI Worldwide Inc. time series problem is compared to works in literature in Table IX. The hybrid method (NSL) has outperformed all the methods and had similar performance as for MO-CCFNN-T=3 in terms of RMSE. Better and stable performance has been achieved by

TABLE VIII
SUNSPOT TIME SERIES RESULTS OF NSL BEING COMPARED WITH LITERATURE

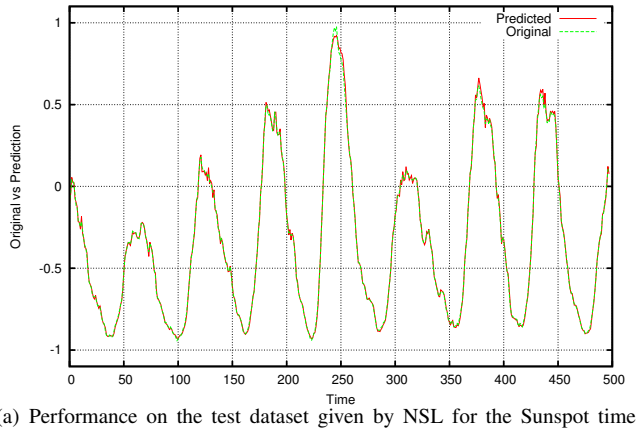
Prediction Method	RMSE	NMSE
RBF with orthogonal least squares (2006) [26]		4.60E-02
Locally linear neuro-fuzzy model (2006) [26]		3.20E-02
SL-CCRNN [13]	1.66E-02	1.47E-03
NL-CCRNN [13]	2.60E-02	3.62E-03
CICC-RNN [18]	1.57E-02	1.31E-03
FNN-NSL [19]	1.33E-02	5.38E-04
Proposed RNN-NSL	1.60E-02	1.37E-03

TABLE IX
ACI WORLDWIDE INC. TIME SERIES RESULTS OF NSL BEING COMPARED WITH LITERATURE

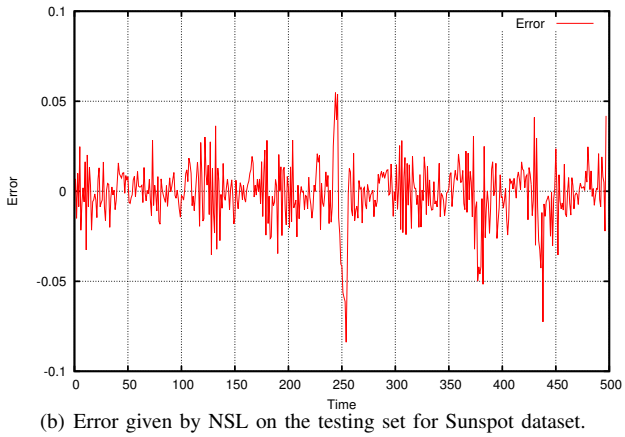
Prediction Method	RMSE	NMSE
CICC-RNN [18]	1.92E-02	
FNN-SL [27]	1.92E-02	
FNN-NL [27]	1.91E-02	
MO-CCFNN-T=2 [28]	1.94E-02	
MO-CCFNN-T=3 [28]	1.47E-02	
FNN-NSL [19]	1.51E-02	1.24E-03
Proposed RNN-NSL	1.47E-02	2.06E-03

TABLE X
SEAGATE TIME SERIES RESULTS OF NSL BEING COMPARED WITH LITERATURE

Prediction Method	RMSE	NMSE
FNN-SL [27]	3.74E-02	
FNN-NL [27]	2.24E-02	
FNN-NSL [19]	2.45E-02	3.56E-03
Proposed RNN-NSL	2.82E-02	8.28E-03

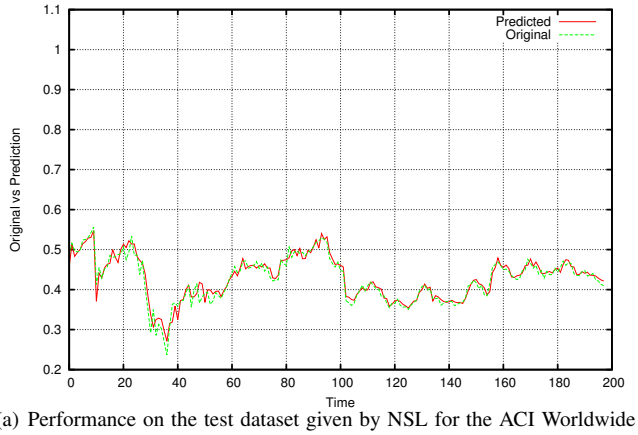


(a) Performance on the test dataset given by NSL for the Sunspot time series.

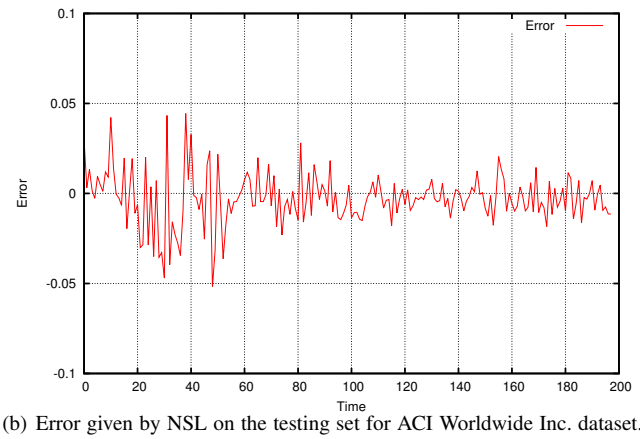


(b) Error given by NSL on the testing set for Sunspot dataset.

Fig. 4. Typical prediction on test dataset given by NSL for Sunspot dataset.

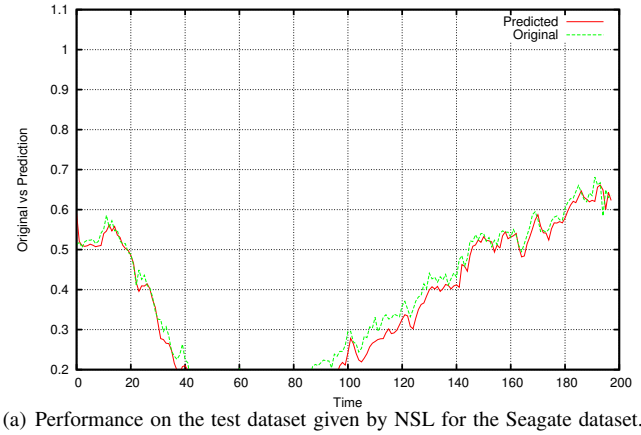


(a) Performance on the test dataset given by NSL for the ACI Worldwide Inc. dataset.

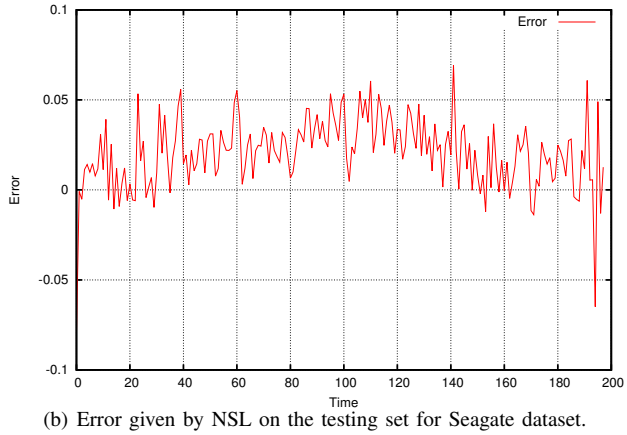


(b) Error given by NSL on the testing set for ACI Worldwide Inc. dataset.

Fig. 5. Typical prediction on test dataset given by NSL for ACI Worldwide Inc. dataset.



(a) Performance on the test dataset given by NSL for the Seagate dataset.



(b) Error given by NSL on the testing set for Seagate dataset.

Fig. 6. Typical prediction on test dataset given by NSL for Seagate dataset.

the proposed method.

The best result of the Seagate time series problem is compared to works in literature in Table X. It has been seen that the NSL method outperformed only FNN-SL method and was unable to beat any other methods it is compared to.

IV. Discussion

The results are competitive for all the different data sets for the modified hybrid model when compared to works from literature. The number of subcomponents and its subpopulation plays a vital role in time series prediction.

The modified hybrid method has given good performances in some of the datasets used when compared to feed forward version of NSL and also when compared to Locally linear neuro-fuzzy model, and competitive cooperative coevolution (CICC) methods.

When compared to standalone methods based on single problem decomposition method (Synapse Level and Neuron Level), NSL gave better performance in two of the cases. Competitive results were even seen in comparison to competitive model (CICC-RNN) based methods in the literature. Better performance was also seen by the proposed method in comparison to adaptive modularity cooperative coevolution (AMCC) on Mackey-Glass data set, where with the given time the problem decomposition method varied.

The limitation in proposed method lies in using single subpopulation for subcomponents from hidden to the output layer since the synapse level decomposition is used in that region. The results of proposed method were unable to beat results of majority datasets when compared to some methods. By increasing the number of sub-populations associated within synapse level decomposition can allow seeing how that region reacts to the increase.

By running each function evaluation on a separate thread (multi-threaded environment) for the modified proposed method (NSL), the computation time can be reduced.

One of the advantages of the proposed hybrid method (NSL) is the use of two problem decomposition. The combination of two problem decomposition (NL and SL) in NSL, allows NL to be used for decision making and SL for diversity in the search. Therefore, NSL performs better than other methods in

some of the cases. The cases where the method was unable to perform is due to either over training or over fitting.

V. Conclusions

This paper has used a modified version of recent problem decomposition method called Neuron-Synapse Level (NSL) for the cooperative coevolution based on recurrent neural networks. NSL has been applied to time series prediction to predict on benchmark dataset and financial dataset. As for the proposed method, it creates more subcomponents when compared to Neuron level and fewer subcomponents when compared to Synapse Level as seen in its earlier version based on feed forward network.

In terms of generalization performance, NSL has been able to get the similar solution quality when compared to other methods it is being compared to. In general, NSL has shown better performance in training than its earlier version based on feedforward network on Mackey-Glass and ACI Worldwide Inc. datasets.

In future work, the proposed method can be applied to pattern classification problems as well as can be used together with feedforward network version to form a competitive model.

References

[1] W. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943. [Online]. Available: http://dx.doi.org/10.1007/BF02478259

[2] M. Paliwal and U. A. Kumar, "Neural networks and statistical techniques: A review of applications," *Expert Systems with Applications*, vol. 36, no. 1, pp. 2 – 17, 2009.

[3] S. Haykin, *Neural Networks and Learning Machines*. Prentice Hall, 2008.

[4] A. Schfer and H. Zimmermann, "Recurrent neural networks are universal approximators," in *Artificial Neural Networks ICANN 2006*, ser. Lecture Notes in Computer Science, S. Kollias, A. Stafylopatis, W. Duch, and E. Oja, Eds. Springer Berlin Heidelberg, 2006, vol. 4131, pp. 632–640.

[5] R. Chandra, "Problem decomposition and adaptation in cooperative neuro-evolution," 2012.

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

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

[8] E. Lorenz, *The Essence of Chaos*. University of Washington Press, 1993.

[9] X. Liang, R. Chen, Y. He, and Y. Chen, "Associating stock prices with web financial information time series based on support vector regression," *Neurocomputing*, vol. 115, pp. 142–149, 2013.

[10] 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.

[11] J. Lehman and R. Miikkulainen, "Neuroevolution," vol. 8, no. 6, p. 30977, 2013.

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

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

[14] 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.

[15] ——, "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, 2012.

[16] R. Chandra, "Adaptive problem decomposition in cooperative coevolution of recurrent networks for time series prediction," in *International Joint Conference on Neural Networks (IJCNN)*, Dallas, TX, USA, Aug. 2013, pp. 1–8.

[17] 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.

[18] 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.

[19] R. Nand and R. Chandra, "Neuron-synapse level problem decomposition method for cooperative neuro-evolution of feedforward networks for time series prediction," in *Neural Information Processing*. Springer, 2015, pp. 90–100.

[20] 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.

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

[22] 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.

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

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

[25] "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

[26] 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.

[27] 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.

[28] ——, "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.