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

Ravneil Nand, Emmenual Reddy, and Mohammed Naseem

School of Computing Information and Mathematical Sciences, University of South Pacific, Suva, Fiji.

Abstract. The breaking down of a particular problem through problem decomposition has enabled complex problems to be solved efficiently. The two major problem decomposition methods used in cooperative coevolution are synapse and neuron level. The combination of both the problem decomposition as a hybrid problem decomposition has been seen applied in time series prediction. The different problem decomposition methods applied at particular area of a network can share its strengths to solve the problem better, which forms the major motivation. In this paper, we are proposing a problem decomposition method that combines neuron and network level problem decompositions for Elman recurrent neural networks and applied to time series prediction. The results reveal that the proposed method has got better results in few datasets when compared to two popular standalone methods. The results are better in selected cases for proposed method when compared to several other approaches from the literature.

Keywords: Cooperative coevolution, problem decomposition, recurrent network, time series prediction

1 Introduction

In the context of evolutionary computation, cooperative coevolutionary algorithms are gaining increasing attention of most researchers. Cooperative coevolution is an evolutionary method that computes the solution of a large problem by decomposing it into subcomponents and solving them individually [1, 2]. An important characteristic of cooperative coevolution is that it offers better diversity using several subcomponents than mainstream evolutionary algorithms [3].

One of the applications of cooperative coevolution have been in the area of neuro-evolution, which include training recurrent neural networks [4, 5]. In a recurrent neural network the current network state and input determines the next state and output. Recurrent neural networks trained using cooperative coevolutionary algorithms are manifested to perform very well for time series prediction problems [3, 6].

Problem decomposition is an important procedure that determines how a neural network is decomposed and encoded as subcomponents [3]. Neuron level and synapse level are the two preeminent problem decomposition methodologies. In neuron level, each neuron in the hidden layer acts as the main reference point for the subcomponents. In synapse level problem decomposition, the network is divided into its lowest level with each weight or synapse in the network forming a subcomponent [2, 7]. Successful application of both decomposition as a hybrid has been seen in [7]. This forms the major motivation for the paper to combine different decomposition methods for successful application to time series prediction.

Time-series analysis is a practicable measure used in various application areas such as the economic models and medical systems. For instance, the prediction of financial markets or the monitoring of physiological signals stemming from a patient during surgery. A crucial objective of time-series analysis is to predict the future behavior of a measurement signal based on its past behavioural observations.

In this paper, we combine neuron and network level problem decomposition to form a new hybrid problem decomposition called Neuron-Network Level (NNL) problem decomposition. NNL is intended for training Elman recurrent neural networks that are chaotic time series problems.

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 dataset overview and experimental setup. In Section 4 and 5, the results and discussion are given respectively. Section 6 concludes the paper with a discussion of future extensions of the paper.

2 Neuron-Network Level Problem Decomposition

In neural networks, Neuron level's emphasis is given on the interacting variables while Network level encoding treats the entire problem as one subset. The problem in cooperative coevolution is to have interacting variables with diversity.

Therefore, we propose a hybrid problem decomposition strategy combining Neuron with Network level decomposition called Neuron-Network level (NNL). NNL breaks down the larger problem into a lower level as in Neuron level problem decomposition. In the proposed method, the subcomponent consists of incoming links associated with neurons in the hidden layer and incoming links to hidden from context layer and one subcomponent for hidden to output layer plus bias as shown in Fig.1. The total of all the outputs generated at each neuron gives the calculation of the actual output as in all the methods mentioned earlier.

The proposed method seems same as Neuron level but the difference lies in the hidden to output layer. If a problem has more than one output than the proposed method will still have one subcomponent for that area whereas Neuron level will depend on the number of output neurons. In addition, the proposed method includes the total number of bias added to the population size whereas neuron level only includes the number of output bias to the population size.

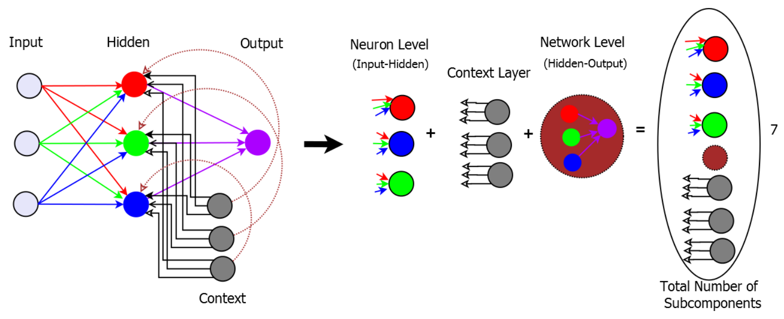


Fig. 1. Neuron-Network Level Problem Decomposition (NNL) breaks down the neural network into specific regions and encodes it into the respective sub-populations.

The total number of sub-components is equal to the total number of hidden layer neurons plus total number on context layer neurons and plus one. The sub-components are implemented as sub-populations.

Algorithm 1: NNL for Training Recurrent Networks

```
Step 1: Decompose the problem into subcomponents according to NNL. Here
  problem is decomposed into input-hidden, context, and hidden-output layer
  subcomponents.
Step 2: Encode each subcomponent in a sub-population according to problem
  decomposition method used.
Step 3: Initialize and cooperatively evaluate each sub-population.
foreach Cycle until termination do
    foreach Sub-population do
        foreach Depth of $n$ Generations do
            Select and create new offspring using genetic operators
            Cooperative Evaluation the new offspring
            Add new offspring's to the sub-population
        end
    end
end
```

Algorithm 1 shows the proposed method which is used in training the recurrent network. In the first phase of the algorithm, the problem is broken down in the number of subcomponents according to the proposed NNL method. In the second phase, the problem is encoded based on the number of neurons in the hidden layer and context layer.

In the third phase, random values are used to initialize all the individuals within the population. Each of the individuals fitness is then evaluated based on the selection of arbitrary individuals existing in the rest of the sub-populations.

Hereafter, using genetic operators the evolution for each sub-population also occurs. Each member of the subpopulations fitness evaluation is done cooperatively with the fittest individuals from the other subsets [8].

Evolution of each sub-population is done for a fixed number of generations and one cycle is accomplished when evolution of all sub-populations have been successfully completed. Evolution of each sub-population is done using the generalized generation gap parent centric crossover (G3-PCX) genetic algorithm [9]. This is where the selected parent is having the best fitness while the rest are selected randomly from the subpopulation to generate the new individuals.

In the next phase the fitness of the new individuals are evaluated. At this point, the random selection of two new parents from the main sub-population is carried out for the purpose of comparison. In future, the two new parents can be replaced by new individuals. In the evolution phase, the fitness evaluation of each individual is carried out together with the fittest individuals from the rest of the sub-populations.

The fitness evaluation of any member of the sub-population is done by joining the fittest members from the rest of the sub-populations. The evolution of these sub-populations is carried out for a fixed number of generations. As soon as maximum fitness evaluations of 50,000 are reached for the network, the generalization performance testing begins.

3 Experimental Setup

The following section gives the overview and setup of the dataset.

Reconstructing the dataset before it can be used is allowed by Taken's embedding theorem [10]. The reconstruction of the chaotic time series data into a state space vector is carried out with the two important conditions of *time delay (T)* and *embedding dimension (D)* [10].

Four different datasets are used to test the proposed method. The *Lorenz time series* is the first data set that is used [11]. It is a simulated time series and is chaotic in nature. To train the neural network, the first 500 values are used whereas the remaining 500 values are used for testing.

The *Sunspot time series* is the second dataset that is used [12]. It is a real world problem time series [12]. This dataset contains 2000 points. To train the neural network, the first 1000 values are used whereas the remaining 1000 values are used for training.

The *ACI Worldwide Inc. time series* is the third data set that is used [13]. To obtain the ACI Worldwide Inc. financial time series data set, the NASDAQ stock exchange is used [13]. To train the neural network, the first 400 values are used whereas to test the neural network the remaining 400 values are used.

The Seagate Technology PLC is the last dataset that is used [13]. It is also a financial time series dataset and contains daily closing stock prices from December 2006 to February 2010. To train the neural network, the first 400 values are used whereas to test the neural network the remaining 400 values are used.

The embedding dimension $D = 3$ and $T = 2$ is used to reconstruct the phase space of Lorenz dataset where as $D = 5$ and $T = 2$ is used to reconstruct the phase space of the rest of the dataset.

According to the literature, the scaling of the four time series dataset is done in the range of [0,1] for ACI Worldwide Inc, Seagate and [-1,1] for Sunspot and Lorenz in order to provide a fair comparison.

Sigmoid units are employed by the recurrent neural network for the Seagate, and ACI Worldwide Inc. time series while the hyperbolic tangent unit is used for Lorenz and Sunspot time series in the hidden and output layers. Root mean squared error (RMSE) is used in evaluating the performance of the proposed method, as done in [3].

The algorithm is run 50 times. For terminating each run of the algorithm, the maximum number of function evaluations was set at 50 000. According to literature [3], a pool size of 2 parents and 2 offsprings are put in the G3-PCX algorithm. For evolving all the sub-populations in the proposed method, the G3-PCX evolutionary model which uses the *generation gap model* [9] for selection is used since it has shown good results with cooperative neuro-evolution [14].

For dividing the problem into subcomponents, the Neuron level and the Synapse level problem decomposition methods are used for comparison with the proposed method since these two methods are established ones. For each sub-population, the number of generations known as *depth of search* was kept at 1 since this configuration has given optimal results in similar work by [3].

To compute the prediction performance of the proposed method, the Root Mean Squared Error (RMSE) is used, as done in [7]. The Equation 1 is used.

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

where y_i, is observed data and $\hat{y}_i$ is predicted data. And N is the observed data's length. This performance measure is used to compare the results with the literature.

4 Results

The experimental results based on the performance of proposed method is given in this section.

The Tables 1 - 4 showcases the results for different number of hidden neurons on the proposed method NNL, being compared to two standalone (NL and SL) methods. The results given in the Tables 1 - 4 are based on 95 percent confidence interval on RMSE and the best results for each algorithm are highlighted in bold. The *Training* shows the train average with train error sum while *Generalisation* is based on test average with test error sum and lastly *Best* shows the best test rmse.

In Table 1, the Lorenz time series problem is been evaluated. It was observed that the NNL has poor performance than the other two methods. It was seen that the generalization performance and training of the NNL and the other two methods increases when the number of the hidden neuron increases. Seven hidden neurons for NNL gave the best result.

Table 1. The prediction training and generalization performance (RMSE) of NL, SL and NNL for the Lorenz time series

Method	H	Training	Generalization	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-NNL	3	0.0369 ± 0.0041	0.0374 ± 0.0041	0.015
	5	0.0357 ± 0.0100	0.0358 ± 0.0099	0.009
	7	**0.0223 ± 0.0023**	**0.0226 ± 0.0024**	**0.007**

Table 2 illustrates the evaluation of the Sunspot time series problem. In this time series, noise is present since it is real-world time series. The proposed method NNL was unable to outperform both of the methods it is compared to but was close in terms of generalization. The best result for NNL was given by seven hidden neurons and even for the other two methods it was seven hidden neurons.

Table 2. The prediction training and generalization performance (RMSE) of NL, SL and NNL for the Sunspot time series

Method	H	Training	Generalization	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-NNL	3	0.0332 ± 0.0064	0.0682 ± 0.0132	0.026
	5	0.0246 ± 0.0024	0.0772 ± 0.0184	0.021
	7	**0.0203 ± 0.0022**	**0.0770 ± 0.0238**	**0.019**

In Table 3, the ACI time series problem results are reported. The time series is real time series where noise is present as in Sunspot time series. For this time series, the NNL didn't perform better than NL method but had similar performance to SL method in terms of generalization. Seven hidden neurons have given the best result for the proposed method.

The Seagate time series problem is evaluated in Table 4. For this time series, the NNL method outperformed the other two methods, NL, and SL. For NNL, seven hidden neurons have given best results.

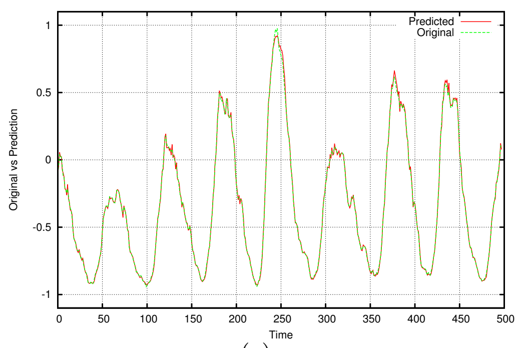
Table 3. The prediction training and generalization performance (RMSE) of NL, SL and NNL for the ACI Worldwide Inc. time series

Method	H	Training	Generalization	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-NNL	3	0.0255 ± 0.0009	0.0212 ± 0.0014	0.015
	5	0.0297 ± 0.0027	0.0222 ± 0.0020	**0.014**
	7	**0.0220** ± **0.0008**	**0.0171** ± **0.0009**	0.015

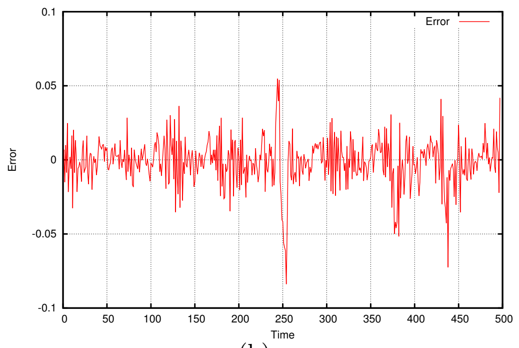
Table 4. The prediction training and generalization performance (RMSE) of NL, SL and NNL for the Seagate time series

Method	H	Training	Generalization	Best
RNN-NL	3	0.0351 ± 0.0549	0.3220 ± 0.0650	0.032
	5	0.0351 ± 0.0448	0.3223 ± 0.0742	0.032
	7	**0.0351±0.0226**	**0.3215±0.0551**	**0.032**
RNN-SL	3	0.0351 ± 0.0628	0.3216 ± 0.0653	**0.031**
	5	0.0351 ± 0.0532	**0.3215±0.0562**	0.032
	7	**0.0350±0.0480**	0.3217 ± 0.0654	0.032
RNN-NNL	3	0.0234 ± 0.0015	0.1841 ± 0.0395	0.030
	5	0.0241 ± 0.0038	0.2121 ± 0.0495	0.049
	7	**0.0199±0.0004**	**0.1764±0.0333**	**0.021**

Figure 2 shows the predicted results of RNN-NNL method with original results on Sunspot dataset. The error graph is also given which indicates that



(a)



(b)

Fig. 2. Typical test dataset prediction by NNL for Sunspot dataset. (a) Performance given by NNL on the testing set. (b) Error on the test dataset given by NNL.

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

Problem	Prediction Method	RMSE	NMSE
Lorenz	RBF with orthogonal least squares (2006) [15]		1.41E-09
	Locally linear neuro-fuzzy model (2006) [15]		9.80E-10
	SL-CCRNN [3]	6.36E-03	7.72E-04
	NL-CCRNN [3]	8.20E-03	1.28E-03
	FNN-NSL [7]	2.34E-03	2.87E-05
	Proposed RNN-NNL	7.49E-03	1.09E-03
Sunspot	RBF with orthogonal least squares (2006) [15]		4.60E-02
	Locally linear neuro-fuzzy model (2006) [15]		3.20E-02
	SL-CCRNN [3]	1.66E-02	1.47E-03
	NL-CCRNN [3]	2.60E-02	3.62E-03
	FNN-NSL [7]	1.33E-02	5.38E-04
	Proposed RNN-NNL	1.89E-02	1.90E-03
ACI	FNN-SL [16]	1.92E-02	
Worldwide	FNN-NL [16]	1.91E-02	
	MO-CCFNN-T=2 [17]	1.94E-02	
	MO-CCFNN-T=3 [17]	1.47E-02	
	Neuron-Synapse Level method FNN-NSL [7]	1.51E-02	1.24E-03
	Proposed NNL	1.44E-02	1.98E-03
Seagate	FNN-SL [16]	3.74E-02	
	FNN-NL [16]	2.24E-02	
	Neuron-Synapse Level methodFNN-NSL [7]	2.45E-02	3.56E-03
	Proposed NNL	2.10E-02	4.58E-03

better methods will be needed to cater for the chaotic nature of the time series problems at certain time intervals.

The best results from Table 1 - 4 with some of the related methods in literature are given in table 5. The RMSE best run together with NMSE are given for comparison purposes. The proposed NNL method has shown good performance in some of the datasets when compared to other methods in the literature.

In Table 5 under problem Lorenz, it shows the best result of Lorenz time series problem being compared to works in literature. It has been seen that the proposed method outperformed only NL-CCRNN method and was close to SL-CCRNN method.

In Table 5 under problem Sunspot, the best result of the Sunspot time series problem is compared with results in the literature where the proposed method has shown to outperform the rest of the methods expect for SL-CCRNN and FNN-NSL methods. The method has given competitive results.

The best result of the ACI Worldwide Inc. time series problem is compared to works in literature in Table 5 under problem ACI. The proposed hybrid method has outperformed all the methods expect for FNN-NSL methods in terms of NMSE. Better and stable performance has been achieved by the NNL.

In Table 5, the best result of the Seagate time series problem is compared with results in the literature under problem Seagate. The proposed method has outperformed all the methods expect for FNN-NSL methods in terms of NMSE. The method had similar performance as in ACI Worldwide Inc. dataset.

5 Discussion

The results obtained for the proposed method is competitive when compared to works from literature involving four different data sets. The application of different decomposition method at different stages of network helps in prediction.

The proposed hybrid model has given better performances in some of the datasets used when compared to FNN-NSL. The results for NNL method on financial datsets are better in terms of RMSE.

In some cases, NNL gave better performance than standalone methods based on cooperative coevolution (CCRNN-Synapse Level and CCRNN-Neuron Level).

The limitation in proposed method lies in the area of hidden to the output layer. The context layer can be decomposed by some other decomposition method to enhance the learning. The results of proposed method was unable to beat results of majority datasets when compared to FNN-NL and FNN-NSL. By increasing and decreasing the number of sub-populations associated within the context layer can allow seeing how that region reacts.

One of the advantages of the proposed hybrid method (NNL) is the use of two problem decomposition. The combination of two problem decomposition (NL and NetL) in NNL, allows NL to be used for decision making and NetL for diversity in the search. Therefore, NNL 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.

6 Conclusion

This paper has applied a new problem decomposition method (NNL) based on Neuron and Network level problem decomposition for the cooperative coevolution to recurrent neural networks. NNL has been tested on time series prediction.

The investigation began with the testing of the proposed hybrid method with benchmark dataset and later on the financial dataset. As for the hybrid model, it creates fewer subcomponents for the problem when compared to Synapse Level and Neuron level.

The results showed that NNL has been able to achieve the similar solution quality when compared to other methods in terms the generalization performance. In general, NNL has shown better optimization performance in time and success rate than other methods on financial datasets.

In future work, the proposed method can be applied to pattern classification problems and global optimization problems.

References

1. N. García-Pedrajas, E. Sanz-Tapia, D. Ortiz-Boyer, and C. Hervás-Martínez, "Introducing multi-objective optimization in cooperative coevolution of neural networks," in *Connectionist Models of Neurons, Learning Processes, and Artificial Intelligence.* Springer, 2001, pp. 645–652.
2. R. Chandra, "Problem decomposition and adaptation in cooperative neuro-evolution," 2012.
3. R. Chandra and M. Zhang, "Cooperative coevolution of Elman recurrent neural networks for chaotic time series prediction," *Neurocomputing*, vol. 186, pp. 116 – 123, 2012.
4. F. Gomez and R. Mikkulainen, "Incremental evolution of complex general behavior," *Adapt. Behav.*, vol. 5, no. 3-4, pp. 317–342, 1997.
5. R. Chandra, M. Frean, and M. Zhang, "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.
6. S. Lawrence, "Noisy time series prediction using a recurrent neural network and grammatical inference."
7. 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.
8. 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.
9. 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.
10. F. Takens, "Detecting strange attractors in turbulence," in *Dynamical Systems and Turbulence, Warwick 1980*, ser. Lecture Notes in Mathematics, 1981, pp. 366–381.
11. E. Lorenz, "Deterministic non-periodic flows," *Journal of Atmospheric Science*, vol. 20, pp. 267 – 285, 1963.
12. S. S., "Solar cycle forecasting: A nonlinear dynamics approach," *Astronomy and Astrophysics*, vol. 377, pp. 312–320, 2001.
13. "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
14. R. Chandra, "Problem decomposition and adaptation in cooperative neuro-evolution," 2012.
15. 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.
16. 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.
17. ——, "Multi-objective cooperative coevolution of neural networks for time series prediction," in *2014 International Joint Conference on Neural Networks (IJCNN)*, July 2014, pp. 190–197.