Competitive Two-Island Cooperative Coevolution for training Elman Recurrent Networks for Time Series Prediction

Rohitash Chandra

Abstract— **Problem decomposition is an important aspect in using cooperative coevolution for neuro-evolution. Cooperative coevolution employs different problem decomposition methods to decompose the neural network training problem into subcomponents. Different problem decomposition methods have features that are helpful at different stages in the evolutionary process. Adaptation, collaboration and competition are characteristics that are needed for cooperative coevolution as multiple sub-populations are used to represent the problem. It is important to add collaboration and competition in cooperative coevolution. This paper presents a competitive two-island cooperative coevolution method for training recurrent neural networks on chaotic time series problems. Neural level and Synapse level problem decomposition is used in each of the islands. The results show improvement in performance when compared to standalone cooperative coevolution and other methods from literature.**

I. Introduction

COOPERATIVE COEVOLUTION (CC) is a nature inspired optimisation method that divides a problem into subcomponents that are similar to the different species in nature [1]. Problem decomposition is an important procedure in cooperation coevolution that determines how the subcomponents are decomposed in terms of their size and the representation of the problem. The original cooperative coevolution method decomposed problems by having a separate subcomponent for each variable [1] and it was later found that the strategy was mostly effective for fully separable problems [2]. Cooperative coevolution naturally appeals to separable problems as there is little interaction among the subcomponents during evolution [3]. In the case of using cooperative coevolution for training neural networks, the problem decomposition method is dependent on the neural network architecture and the type of the training problem [4] in terms of the level of inter-dependencies amongst the neural network weights.

The two major problem decomposition methods are those on the synapse level [5] and neuron level [6], [7]. Different problem decomposition methods have shown different level of strengths and weaknesses in different types of problems and neural network architectures. Neural level problem decomposition methods have shown good performance in pattern classification problems [8], [9], [6], [7] while synapse level problem decomposition have shown good performance in control and time series prediction problems [5], [10], [11].

Competition is a major feature in biological evolution. The initial motivations for using competition in evolutionary algorithms has been given by Rosin and Belew [12]. They presented a competitive coevolution method where a population called "host" and another called "parasite" compete with each other with different mechanisms that enable fitness sharing, elitism and selection. In cooperative coevolution, competition has been used for multi-objective optimisation [13] that exploited correlation and inter-dependencies between the components of the problem. Competition has also been used in cooperative coevolution based multi-objective optimisation in dynamic environments where problem decomposition method adapts according to the change of environment rather than being static from the beginning of the evolution. [14].

Adaptation of problem decomposition in different phases of evolution has been effective for training feedforward networks on pattern recognition problems [15] and recurrent networks on grammatical inference problems [16]. The results have shown that it is reasonable to adapt the problem decomposition method at different stages of evolution. However, adaptation of problem decomposition method at different stages of evolution is costly in terms of parameter setting. It is difficult to establish the right parameters that indicates when to switch from one problem decomposition to another and how long to use them at the different stages of evolution [16]. Extensive experiments are needed when adaptation of problem decomposition is applied to different neural network architectures and problems.

This paper presents a new approach to neuro-evolution of recurrent neural networks using cooperative coevolution that enforces competition using different problem decomposition methods. The proposed approach takes advantage of the different problem decomposition methods that compete and collaborate with each other with the exchange of the strongest genetic materials during evolution. The approach is applied to chaotic time series problems using Elman recurrent neural networks [17]. The performance of the proposed approach is compared with established problem decomposition methods from literature along with other computational intelligence methods.

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

Dr. Rohitash Chandra is with the School of Computing, Information and Mathematical Sciences, University of the South Pacific, Laucala Campus, Suva, Fiji.

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 [1]. 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 [1]. It was later found that this strategy is only effective for problems which are fully separable [2]. 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 [2], [18], [19], [20].

A function of n variables is separable if it can be written as a sum of n functions with just one variable [21]. Non-separable problems have inter-dependencies 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 [4].

The sub-populations 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 problem decomposition method has been able to group the interacting variables into separate subcomponents [7]. 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 [4].

Cooperative coevolution has been been used for neuro-evolution of recurrent neural networks for time series problems [11], [22] and it has been shown that the perform better compared to several methods from literature.

B. Recurrent Neural Network

Recurrent neural networks have been an important focus of research as they can be applied to difficult problems involving time-varying patterns. They are suitable for modeling temporal sequences. First-order recurrent neural networks use context units to store the output of the state neurons from computation of the previous time steps. The context layer is used for computation of present states as they contain information about the previous states. The Elman architecture [17] employs a context layer which makes a copy of the hidden layer outputs in the previous time steps. The dynamics of the change of hidden state neuron activation's in Elman style recurrent networks is given by Equation (1).

$$y_i(t) = f\left(\sum_{k=1}^{K} v_{ik}\ y_k(t-1) + \sum_{j=1}^{J} w_{ij}\ x_j(t-1)\right) \quad (1)$$

where $y_k(t)$ and $x_j(t)$ represent the output of the context state neuron and input neurons respectively. v_{ik} and w_{ij} represent their corresponding weights. $f(.)$ is a sigmoid transfer function.

C. Problem Decomposition for Recurrent Networks

Problem decomposition is an important procedure in using cooperative coevolution for neuro-evolution. The problem decomposition method will determine which set of weights in from the neural network will be encoded into a particular sub-population of cooperative coevolution. In the case of recurrent neural networks, special consideration needs to be made for the weights that are associated with the feedback connections.

There are two major problem decomposition methods for neuro-evolution that decomposes the network on the *neuron level* 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 [5] and neural fuzzy network with cultural cooperative particle swarm optimisation [10]. In neuron level problem decomposition, the neurons in the network act as the reference point for the decomposition. Examples include enforced sub-populations [8], [9] and neuron-based sub-population [6], [7].

III. Competition and Collaboration in Cooperative Coevolution

Competition in an environment of limited resources is an important feature used for survival in nature. Collaboration helps in the sharing of resources between the different species that have different characteristics for adaption when given with environmental changes and other challenges. In cooperative coevolution, the species are implemented as sub-populations that do not exchange genetic material with other sub-populations. Collaborations and exchange of genetic material or information between the sub-populations can be helpful in the evolutionary process. Competition and collaboration is vital component of evolution where different groups of species compete for resources in the same environment. Different types of problem decomposition methods in cooperative coevolution represent different groups of species (neuron and synapse level [5], [6], [7]) in an environment that features collaboration through fitness evaluation during evolution.

In this section, we propose a cooperative coevolution method that incorporates competition and collaboration with species that is motivated by evolution in nature. The proposed method employs the strength of a different problem decomposition method which reflects on the different degree of non-separability (interaction of variables) and diversity (number of sub-populations) during evolution [4].

The proposed method is called *Competitive Island-Based Cooperative Coevolution* (CICC) that employs different problem decomposition methods that compete with different features they have in terms of diversity and degree of non-separability. In the rest of the discussion, we refer to the different types of problem decomposition as *islands*. The proposed method features competition where the the different islands compare their solutions after a fixed time (number of fitness evaluations) and exchange the best solution between the islands. In this model, for the case of neuro-evolution, only two islands are used as given by the established problem decomposition methods. The details of the different problem decomposition methods that are called *islands* are given below.

1) **Synapse level problem decomposition**: Decomposes the network into its lowest level to form a single subcomponent [5], [10]. The number of connections in the network determines the number of subcomponents.
2) **Neuron level problem decomposition**: Decomposes the network into neuron level. The number of neurons in the hidden, state and output layer determines the number of subcomponents [7].

The proposed CICC method is given in Algorithm 1. Initially, all the sub-populations of the synapse level and neuron level islands are randomly initialised with random real values in a range. In Stage 1, the sub-populations at synapse and neuron level problem decomposition are cooperatively evaluated.

State 2 proceeds with evolution in an island based round-robin fashion where each island is evolved for a predefined time based on the number of fitness evaluations. This is called *island evolution time* that is given by the number of *cycles* that makes the required number of function evaluations in synapse and neuron level islands. A cycle in cooperative co-evolution is when all the sub-populations have been evolved for n number of generations in a round-robin fashion.

Once a particular island has been evolved for the island evolution time, the algorithm proceeds and check if the best solution of the particular island is better than the rest of the islands. If the solution is the best, then the *collaboration* procedure takes place where the solution is copied to the rest of the islands. Afterwards, when present island changes, the best solution competes within the rest of the solutions from same island until the local evolution time has been reached. In the collaboration procedure, the algorithm needs to take into account on how the solution from one island will be transferred into the other the island which is defined by a different problem decomposition method.

A. Cooperative Evaluation

Cooperative evaluation of individuals in the respective sub-populations is done by concatenating the chosen individual from a given sub-population with the best individuals from the rest of the sub-populations [1], [6], [7], [11]. The concatenated individual is encoded into the recurrent neural network and the fitness is calculated. 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.

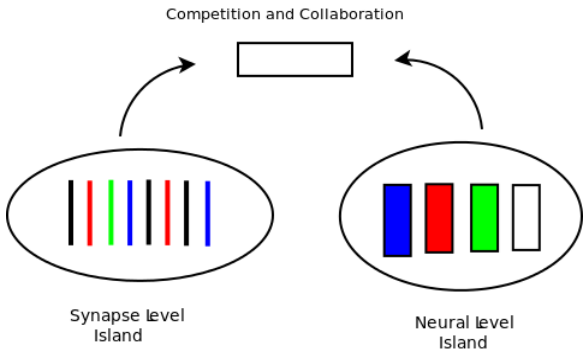

Fig. 1. The Two-Island CICC method that employs neuron and synapse level islands.

Alg. 1 Competitive Two-Island Cooperative Coevolution for training Recurrent Neural Networks

Stage 1: Initialisation:

i. Cooperatively evaluate Neuron level
ii. Evaluate Network level

Stage 2: Evolution:

while *FuncEval* $\leq$ *GlobalEvolutionTime* **do**
  while *FuncEval* $\leq$ *Island-Evolution-Time* **do**
    foreach *Sub-population at Synapse level* **do**
      foreach *Depth of* n *Generations* **do**
        Create new individuals using genetic operators
        Cooperative Evaluation
      end
    end
  end
  while *FuncEval* $\leq$ *Island-Evolution-Time* **do**
    foreach *Sub-population at Neuron level* **do**
      foreach *Depth of* n *Generations* **do**
        Create new individuals using genetic operators
        Cooperative Evaluation
      end
    end
  end
  Stage 2: Competition: Compare and mark the island with best fitness.

  Stage 3: Collaboration: Inject the best individual from the island with better fitness into the other island.
end

B. Competition

Each island employs a different problem decomposition method. In the synapse level island, a much higher number of function evaluation is required for a single cycle when compared to the neuron level island. The number of function evaluation depends on the number of sub-populations used in the island. Synapse level island employs the highest number of sub-populations as each weight link is represented as a

sub-population, whereas, neuron level sub-populations have more than one weight variables.

Both islands need to be given same time for evolution, therefore, the number of function evaluations required need to be the same or similar. We can only evolve each island for complete cycles, therefore, the number of function evaluations cannot be exactly the same for each island. In the competitive framework, both islands are given similar approximate time in terms of the number of function evaluations.

C. Collaboration

After the competition, the island that contains an individual with the best solution is then injected (copied) into the other islands. A number of factors needs to be taken into account when making such a transfer as the size and number of subcomponents vary for each island due to their difference in problem decomposition method. The best individuals from each of the subcomponents needs to be carefully concatenated into an individual and transferred without losing any genotype (subcomponents in cooperative coevolution) to phenotype (recurrent neural network) mapping.

Once the transfer is done, both islands need to be evaluated in order to prepare for the next round of competition. In evaluation, the best individuals in each of the sub-populations are marked for evolution and cooperative evaluation. At each round, only the winner island is not evaluated.

The winner island is used to inject the best solution to the other island. The island in which the best individual is injected is evaluated to ensure that the injected individual has a fitness. In order to save evaluation time, the fitness can also be transferred along with the solution. This depends on the way the sub-populations are implemented and the approach taken in ensuring that the fitness value is updated at the right position that corresponds with the individual that has been transferred. Since each sub-population contain individuals that have a fitness, we need to note that there will be a number of different fitness values from the best individual in each sub-population. We only take the best fitness value and use it to replace the best individuals from all the sub-populations in the other island. Since the number of sub-populations is different, only the best fitness replace the old best fitness as it carries a stronger solution.

The type of evolutionary algorithm used in the sub-population will have certain requirements for such a transfer of solution to take place. In our implementation, we used the generalised generation gap with parent-centric crossover (G3-PCX) evolutionary algorithm [23] in the sub-populations. This algorithm's selection criteria is the generalised generation gap model where a handful of individuals are replaced at every generation and only those that are replaced are evaluated.

IV. Simulation and Analysis

This section presents an experimental study of competitive island based cooperative coevolution for training recurrent neural networks on chaotic time series problems. The neuron level (NL) [11] and synapse level (SL) [11] problem decomposition methods are used in each of the islands and standalone versions of these methods are used for comparison.

The Mackey Glass time series [24] and Lorenz time series [25] are the two simulated time series while the real-world problems are the Sunspot time series [26] and the financial time series from *ACI Worldwide Inc* given in NASDAQ stock exchange [27].

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 [11]. 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 [28]. 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 [29]. Taken's proved that if the original attractor is of dimension d, then $D = 2d+1$ will be sufficient to reconstruct the attractor [28].

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 [30], [31], [11].

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 2 and Equation 3.

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

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

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

A. Problem description

The *Mackay Glass time series* has been used in literature as a benchmark problem due to its chaotic nature [24]. The differential equation used to generate the Mackey Glass time series is given in Equation 4.

$$\frac{\delta x}{\delta t} = \frac{ax(t-\tau)}{[1 + x^c(t-\tau)]} - bx(t) \tag{4}$$

In Equation 4, the delay parameter τ 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 [32], [33], [34], [35] 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 4. The first 500 samples are used for training the Elman network while rest of the 500 samples are used for testing. The time series is scaled in the range [0,1]. The phase space of the original time series is reconstructed with the embedding dimensions $D = 3$ and $T = 2$.

The *Lorenz time series* was introduced by Edward Lorenz who has extensively contributed to the establishment of Chaos theory [25]. The Lorenz time series is chosen for prediction and 1000 samples are generated from the Lorenz equation [25]. 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 [36]. 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 [26]. 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 [34], [35]. 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.

The *financial time series* dataset is taken from the NASDAQ stock exchange [27]. It contains daily closing prices for ACI Worldwide Inc time series which is one of the companies listed on the NASDAQ stock exchange. The data set contains closing stock prices from December 2006 to February 2010 which is equivalent to around 800 data points and 400 time series points after embedding $T = 2$. We used embedding dimension $D = 5$ to reconstruct the time series data using Taken's theorem in order to get the training and testing data sets. The closing stock prices were normalized between 0 and 1. The dataset also overlaps with the recession that hit the US market. The given data points were divided into training and testing using a 50-50 split.

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 and financial time series while hyperbolic tangent unit is used for Lorenz and Sunspot time series. The experiment set-up is same as our previous works [11]. The RMSE and NMSE given in Equation 2 and Equation 3 are used as the main performance measures of the recurrent network.

In the proposed CICC for recurrent networks shown in Algorithm 1, each sub-population 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 neuron and synapse level decomposition [7]. Hence, 1 is used as the depth of search in all the experiments. Note that all sub-populations evolve for the same depth of search.

The termination condition of the all the problems and recurrent network training methods is when a total of 50 000 function evaluations has been reached by the respective cooperative co-evolutionary methods (CC-NL and CC-SL). The proposed CICC method employs a total of 100 000 function evaluation where each island (SL and NL) employs 50 000 function evaluations.

C. Results and discussion

This section reports the performance of CICC for training the Elman recurrent network on the chaotic time series problems.

The results are given for different number of hidden neurons for Elman style recurrent networks using the respective co-evolutionary algorithms given in Tables I - IV. The CC-NL and CC-SL represent standalone cooperative coevolution neuron and synapse level methods, respectively. They are used to compare with the proposed CICC SL-NL method using the same setup for the recurrent network architecture and optimisation time in terms of function evaluations as given in the experimental set-up subsection.

The results report the RMSE of with mean and 95 percent confidence interval along with the best run from 50 experimental runs.

We evaluate the results by comparing the different methods with the number of hidden neurons (H). Note that the least values of RMSE shows the best results. In Table I, the results of the Mackey Glass time series shows that the CICC method has given better performances than CC-NL and CC-SL. This is clear for all the cases, i.e for 3 - 9 hidden neurons. In Table II, similar trend is seen where CICC outperforms standalone CC-SL and CC-NL. CICC gives better performance as the number of hidden neuron increases for both simulated problems. This is seen for the training, generalisation and the best runs. We note that both of these problems are simulated time series that do not contain noise, hence, there was no problem faced in overfitting that is common for poor generalisation performance.

The results in Table III and Table IV reveal the performance of the proposed method for real world time series where noise is present. In the Sunspot time series, CICC performs better than the other methods for all the cases. In the same trend is seen for the finance time series (ACI

Worldwide Inc), however, the improvement is not that high when compared to the other problems. In both cases, CICC gives better performance as the number of hidden neuron increases.

TABLE I

THE TRAINING AND GENERALISATION PREDICTION PERFORMANCE FOR THE MACKEY-GLASS TIME SERIES

Method	H	Training (×E-02)	Generalisation (×E-02)	Best (×E-02)
CC-NL	3	1.138 ± 0.104	1.143 ±0.105	0.557
	5	1.600 ± 0.111	4.374 ± 1.113	1.239
	7	1.680 ± 0.121	4.339 ± 1.132	1.522
	9	1.776 ± 0.106	6.886 ± 1.792	1.466
CC-SL	3	1.811± 0.208	1.820 ±0.209	0.976
	5	1.636 ±0.192	1.643 ±0.193	0.822
	7	1.906 ±0.414	1.911±0.415	0.902
	9	2.964 ±0.685	2.967± 0.685	1.056
CICC	3	1.053 ±0.063	1.059 ± 0.064	0.572
SL-NL	5	0.847 ±0.062	0.847 ± 0.063	0.460
	7	0.846 ±0.063	0.847 ± 0.064	0.470
	9	0.856 ±0.074	0.858 ±0.074	0.399

TABLE II

THE TRAINING AND GENERALISATION PREDICTION PERFORMANCE FOR THE LORENZ TIME SERIES

PD	H	Training (×E-02)	Generalisation (×E-02)	Best (×E-02)
CC-NL	3	1.775 ± 0.153	1.839 ± 0.161	0.728
	5	1.321 ± 0.143	1.355±0.147	0.319
	7	1.425 ± 0.153	1.470 ±0.156	0.514
	9	1.489 ± 0.159	1.553 ± 0.167	0.514
CC-SL	3	2.085 ± 0.242	2.136± 0.246	0.787
	5	1.678± 0.199	1.748 ± 0.210	0.433
	7	1.643 ± 0.282	1.715 ± 0.296	0.591
	9	1.444 ± 0.191	1.513 ± 0.205	0.642
CICC	3	1.390 ±0.155	1.431 ±0.158	0.583
SL-NL	5	1.026 ±0.136	1.054 ±0.140	0.355
	7	0.938 ±0.150	0.965 ±0.149	0.372
	9	0.888 ±0.097	0.915 ±0.101	0.442

TABLE III

THE TRAINING AND GENERALISATION PREDICTION PERFORMANCE FOR THE SUNSPOT TIME SERIES

PD	H	Training (×E-02)	Generalisation (×E-02)	Best (×E-02)
CC-NL	3	2.066 ± 0.217	5.119 ± 1.233	1.693
	5	1.794 ± 0.187	5.369 ± 1.277	1.662
	7	1.648 ± 0.100	5.656 ± 1.553	1.510
	11	1.705 ± 0.159	6.513 ± 1.890	1.507
CC-SL	3	2.066 ± 0.217	5.119 ± 1.233	1.693
	5	1.794 ± 0.187	5.369 ± 1.277	1.662
	7	1.648 ± 0.100	5.656 ± 1.553	1.510
	11	1.705 ± 0.159	6.513 ± 1.890	1.507
CICC	3	1.589 ±0.090	4.068 ±0.594	1.572
SL-NL	5	1.479 ± 0.108	4.544± 1.229	1.342
	7	1.348 ± 0.0745	7.606 ± 2.219	1.663
	9	1.485 ± 0.071	6.657 ± 1.771	1.778

Tables V - VII, compare the best results from the previous tables with some of the established methods in literature. The RMSE from the best run is used for comparison along with the NMSE that was obtained particularly for comparison of results from the literature. We note that the particular financial time series data-set has not been used in the literature, therefore we cannot provide any comparison

TABLE IV

THE TRAINING AND GENERALISATION PREDICTION PERFORMANCE FOR THE FINANCE (ACI WORLDWIDE INC) TIME SERIES

PD	H	Training (×E-02)	Generalisation (×E-02)	Best (×E-02)
CC-NL	3	2.074 ± 0.041	2.117 ± 0.132	1.934
	5	2.027 ± 0.030	2.041 ± 0.024	1.931
	7	2.010 ± 0.019	2.043 ± 0.044	1.932
	9	2.028 ± 0.019	2.049 ± 0.066	1.930
CC-SL	3	2.262 ± 0.072	2.186 ± 0.078	1.908
	5	2.200 ± 0.074	2.105 ± 0.047	1.930
	7	2.108 ± 0.051	2.108 ± 0.058	1.931
	9	2.106 ± 0.041	2.170 ± 0.065	1.947
CICC	3	2.008 ± 0.027	2.039 ± 0.028	1.920
SL-NL	5	1.974 ± 0.022	2.031 ± 0.019	1.942
	7	1.941 ± 0.018	2.027 ± 0.030	1.935
	9	1.932 ± 0.019	2.005 ± 0.015	1.942

The proposed CICC method has given better performance when compared to 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) [10]. The only exception is being the results from Hybrid NARX-Elman networks [35] as it has additional enhancements such as the optimisation of the embedding dimensions and strength of architectural properties of hybrid neural networks with residual analysis [35].

CICC performs better than standalone cooperative coevolution in literature, (CCRNN-Synapse Level and CCRNN Network Level). It also performs better when compared to adaptive modularity cooperative coevolution (AMCC), where the motivation was to change the problem decomposition method with time, i.e begin with Synapse level and then move to neuron level and network level where only a standard evolutionary algorithm is used. This approach intended to give the appropriate problem decomposition method at different stages of evolution. This approach had limitations due to parameter setting and heuristics required to figure out when to change from one problem decomposition to another ad there is no established measure of the interacting variables as given by the degree of non-separability [4].

Synapse level island would be most useful in separable problems that have lower degree of non-separability - it provides more flexibility and enforces global search through the sub-populations. CICC fulfils the limitations faced by fixed problem decomposition methods by using the best solutions after each round of competition of the islands. In this way, the search can escape from local minimum from the solution from the other island.

CICC can be further improved by adding more islands - which will depend on different problem decomposition methods. Some of the island can also be composed by problem decomposition where the number of sub-population and its composition is chosen arbitrarily.

A major advantage of the proposed method is that it can be implemented in a multi-threaded environment that will speed up the computation time which is a limitation of cooperative

TABLE V

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

Prediction Method	RMSE	NMSE
Neural fuzzy network and hybrid of cultural algorithm and cooperative particle swarm optimisation (CCPSO) (2009) [10]	8.42E-03	
Neural fuzzy network and particle swarm optimisation (PS0) (2009) [10]	2.10E-02	
Neural fuzzy network and cooperative particle swarm optimisation (CPS0) (2009) [10]	1.76E-02	
Neural fuzzy network and differential evolution (DE) (2009) [10]	1.62E-02	
Neural fuzzy network and genetic algorithm (GA) (2009)[10]	1.63E-02	
Backpropagation neural network and genetic algorithms with residual analysis (2011) [37]	1.30E-03	
Hybrid NARX-Elman RNN with Residual Analysis (2010) [35]	3.72E-05	2.70E-08
Backpropagation neural network and genetic algorithms with residual analysis (2011) [37]	1.30E-03	
CCRNN-Synapse Level (2012) [11]	6.33E-03	2.79E-04
CCRNN-Neuron Level (2012) [11]	8.28E-03	4.77E-04
AMCC-RNN [22]	7.53E-03	3.90E-04
Proposed CICC-RNN	3.99E-03	1.11E-04

TABLE VI

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

Prediction Method	RMSE	NMSE
Backpropagation-through-time (BPTT-RNN) (2010) [31]		1.85E-03
Real time recurrent learning (RTRL-RNN) (2010) [31]		1.72E-03
Recursive Bayesian LevenbergMarquardt (RBLM-RNN) (2010) [31]		9.0E-04
Hybrid NARX-Elman RNN with Residual Analysis (2010) [35]	1.08E-04	1.98E-10
Backpropagation neural network and genetic algorithms with residual analysis (2011) [37]	2.96E-02	
CCRNN-Synapse Level (2012) [11]	6.36E-03	7.72E-04
CCRNN-Neuron Level (2012) [11]	8.20E-03	1.28E-03
AMCC-RNN [22]	5.06E-03	4.88E-04
Proposed CICC-RNN	3.55E-03	2.41E-04

TABLE VII

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

Prediction Method	RMSE	NMSE
Multi-layer perceptron (1996) [30]		9.79E-02
Elman RNN (1996) [30]		9.79E-02
FIR Network (MLP) (1996) [30]		2.57E-01
Wavelet packet multilayer perceptron (2001)[38]		1.25E-01
Radial basis network with orthogonal least squares (RBF-OLS)(2006) [34]		4.60E-02
Locally linear neuro-fuzzy model - Locally linear model tree (LLNF-LoLiMot) (2006) [34]		3.20E-02
Hybrid NARX-Elman RNN with Residual Analysis (2010) [35]	1.19E-02	5.90E-04
CCRNN-Synapse Level (2012) [11]	1.66E-02	1.47E-03
CCRNN-Neuron Level (2012) [11]	2.60E-02	3.62E-03
AMCC-RNN [22]	2.41E-02	3.11E-03
Proposed CICC-RNN	1.57E-02	1.31E-03

coevolution for training neural network when compared to gradient based methods. In a multi-threaded implementation, each island can run on a separate thread.

V. CONCLUSIONS AND FUTURE WORK

This paper presented competitive island-based cooperative coevolution of recurrent neural networks for chaotic time series prediction. The proposed method used two different islands that were composed by defined by different problem decomposition methods. The results have shows that the proposed method outperforms the standalone cooperative coevolution methods in terms of prediction performance and scalability. The proposed method also performs better than several other methods from the literature. The proposed method takes advantage of two problem decomposition methods with different degree of non-separability. In a conventional cooperative coevolution method, the problem decomposition method is fixed throughout the evolutionary process, whereas in the proposed approach, two methods compete and collaborate through the islands. In case when the search is trapped in a local minimum in a particular island, the search takes advantage of the solution that is produced in the other island through the collaborative features.

In future work, the proposed method can be improved by exploring other problem decomposition methods that can provide more competition. A multi-threaded version of the algorithm can be developed to reduce the computation time. The method can be used to evolve other neural network architectures for similar problems and those that involve pattern classification and control. The proposed method can

be also use for large scale global optimisation problems.

REFERENCES

[1] M. A. Potter and K. A. De Jong, "A cooperative coevolutionary approach to function optimization," in *PPSN III: Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature*. London, UK: Springer-Verlag, 1994, pp. 249–257.

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

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

[4] 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.

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

[6] R. Chandra, M. Frean, and M. Zhang, "An encoding scheme for cooperative coevolutionary neural networks," in *23rd Australian Joint Conference on Artificial Intelligence*, ser. Lecture Notes in Artificial Intelligence. Adelaide, Australia: Springer-Verlag, 2010, pp. 253–262.

[7] 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.

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

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

[10] C.-J. Lin, C.-H. Chen, and C.-T. Lin, "A hybrid of cooperative particle swarm optimization and cultural algorithm for neural fuzzy networks and its prediction applications," *Trans. Sys. Man Cyber Part C*, vol. 39, pp. 55–68, January 2009.

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

[12] C. Rosin and R. K. Belew, "New methods for competitive coevolution," *Evolutionary Computation*, vol. 5, no. 1, pp. 1 –29, 1997.

[13] C. Goh, K. Tan, D. Liu, and S. Chiam, "A competitive and cooperative co-evolutionary approach to multi-objective particle swarm optimization algorithm design," *European Journal of Operational Research*, vol. 202, no. 1, pp. 42 – 54, 2010.

[14] C.-K. Goh and K. C. Tan, "A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 13, no. 1, pp. 103 –127, feb. 2009.

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

[16] ——, "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.

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

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

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

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

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

[22] R. Chandra, "Adaptive problem decomposition in cooperative coevolution of recurrent networks for time series prediction," in *International Joint Conference on Neural Networks (IJCNN)*, August 2013, p. In Press.

[23] 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.

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

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

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

[27] "NASDAQ Exchange Daily 1970-2010 Open, Close, High, Low and Volume , howpublished = http://www.infochimps.com/datasets/nasdaq-exchange-daily-1970-2010-open-close-high-low-and-volume, note = Accessed: 2013-04-30."

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

[29] 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.

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

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

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

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

[34] 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.

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

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

[37] M. Ardalani-Farsa and S. Zolfaghari, "Residual analysis and combination of embedding theorem and artificial intelligence in chaotic time series forecasting," *Appl. Artif. Intell.*, vol. 25, pp. 45–73, January 2011.

[38] K. K. Teo, L. Wang, and Z. Lin, "Wavelet packet multi-layer perceptron for chaotic time series prediction: Effects of weight initialization," in *Proceedings of the International Conference on Computational Science-Part II*, ser. ICCS '01, 2001, pp. 310–317.