Competition and Collaboration in Cooperative Coevolution of Elman Recurrent Neural Networks for Time-Series Prediction

Rohitash Chandra

***Abstract*— Collaboration enables weak species to survive in an environment where different species compete for limited resources. Cooperative coevolution (CC) is a nature-inspired optimization method that divides a problem into subcomponents and evolves them while genetically isolating them. Problem decomposition is an important aspect in using CC for neuroevolution. CC 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 needed for CC, as multiple subpopulations are used to represent the problem. It is important to add collaboration and competition in CC. This paper presents a competitive CC method for training recurrent neural networks for chaotic time-series prediction. Two different instances of the competitive method are proposed that employs different problem decomposition methods to enforce island-based competition. The results show improvement in the performance of the proposed methods in most cases when compared with standalone CC and other methods from the literature.**

***Index Terms*— Chaotic time series, cooperative coevolution (CC), genetic algorithms, neuroevolution, recurrent neural networks.**

I. Introduction

IN NATURE, competition and collaboration play crucial roles of survival for different species given limited resources. Collaboration enables weak species to survive in an environment where different species compete for limited resources. Different species compete among themselves, and at times, collaborate with other species to exchange resources. Different species have different levels of strengths according to their genes, population diversity, and the environmental conditions. Cooperative coevolution (CC) is a nature-inspired optimization method that divides a problem into subcomponents that are analogous to the different groups of species in nature [1]. Problem decomposition is an important procedure of CC that determines how the subcomponents are decomposed in terms of their size and the portion of the problem that the subcomponents represent.

Manuscript received January 22, 2014; revised July 20, 2014, November 19, 2014, and February 11, 2015; accepted February 13, 2015. Date of publication March 5, 2015; date of current version November 16, 2015.

The author is with the School of Computing, Information and Mathematical Sciences, University of the South Pacific, Suva, Fiji, and also with the Artificial Intelligence and Cybernetics Research Group, Software Foundation, Nausori, Fiji (e-mail: c.rohitash@gmail.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2015.2404823

The original CC 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]. CC naturally appeals to separable problems as there is little interaction among the subcomponents during evolution [3]. In the case of using CC for training neural networks, the problem decomposition method is dependent on the neural network architecture and the type of the training problem in terms of the level of interdependencies among the neural network weights [4].

The two major problem decomposition methods are those at the synapse level (SL) [5] and at the neuron level (NL) [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 [6]–[9], while SL problem decomposition methods 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 have been given in [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 CC, competition has been used for multiobjective optimization [13] that exploited correlation and interdependencies between the components of the problem. Competition has also been used in CC-based multiobjective optimization 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]. Adaptation of problem decomposition method at different stages of evolution is costly as it is difficult to establish optimal parameters that indicate when to switch from one problem decomposition to another and how long to use them [16]. Extensive experiments are needed when adaptation of problem decomposition is applied to different neural network architectures and problems [16].

The strengths for using competition in evolutionary algorithms [12] have given the motivation to incorporate

it in CC, which can be beneficial as different problem decomposition methods can be evolved. Competition can ensure that the different problem decomposition methods are given an opportunity during the entire evolution as opposed to adaptive problem decomposition method as in our previous work [16], [17] where the problem decomposition method is adapted over time. In this way, there is no problem in finding the right problem decomposition method at a particular time according to the degree of separability [4].

This paper presents a new method to neuroevolution of Elman recurrent networks [18] using CC that enforces competition using different problem decomposition methods. Elman recurrent networks employs the context weights that feature information of the past state of the network in order to make future decisions, which is needed for time-series prediction [18]. We present a competitive two-island and three-island CC method for training recurrent neural networks for chaotic time-series problems where a one-step ahead prediction is used. The proposed method takes advantage of different problem decomposition methods that evolve and compete with each other and at the same time collaborate with each other with the exchange of the strongest genetic materials during evolution. NL and SL problem decomposition is used in each of the islands of the respective competitive methods. In the three-island method, network level (NetL) of problem decomposition is used as well. The performance of the proposed approach is compared with established problem decomposition methods from literature along with other computational intelligence methods. This paper extends our previous work [19] where a competitive two-island CC method was proposed. The proposed competitive three-island CC method is used along with the two-island method, and the results are further compared and evaluated.

The rest of this paper is organized as follows. A brief background on time-series prediction, CC, and recurrent neural networks is presented in Section II, and Section III gives details of the competitive and collaborative CC method for training recurrent networks. Section IV presents a background on the given chaotic time-series problems, experimental results, and discussion. Finally, the conclusion is drawn in Section V.

II. Background and Related Work

A. Computational Intelligence and Neuroevolution for Time-Series Prediction

Time-series prediction involves the use of past and present time-series data to make future predictions [20], [21]. The applications for time-series prediction are wide that range from weather prediction [22] to financial prediction [23]–[27].

Computational intelligence methods have been popular in time-series prediction that includes multilayer perceptron [28], Elman recurrent networks [28], radial basis networks, and locally linear neurofuzzy methods [29]. Real-time recurrent learning algorithm and recursive Bayesian network with Levenberg–Marquardt algorithm [30] have also been used and shown promising results. Hybrid Elman–NARX neural networks have been used for chaotic time-series prediction that produced exceptional results with the benchmark datasets [31]. Similar method was also used for backpropagation network with residual analysis that showed competitive results [32]. Type-2 fuzzy neural networks [33] have also been recently proposed for time-series prediction that employs weight update using backpropagation.

Evolutionary computation methods have been used with neural networks and other soft computing methods for time-series prediction [10], [11], [34]. Hybrid of cultural algorithms and cooperative particle swarm optimization (CCPSO) have been proposed for time-series prediction [10]. In real-world applications, evolutionary radial-basis networks have been used for financial time-series prediction on data from the Taiwan Stock Index [25]. Furthermore, the performance of a fuzzy evolutionary and neuroevolutionary feedforward neural networks has been compared for financial time-series problems [27].

The importance for the time lag in time-series problems has been explored where a simple deterministic method was proposed for the selection of optimal time lags for nonuniform embedding [35]. The method is able to handle optimization problems in a multiparameter space of arguments, while improving time-series prediction. Quantum-inspired hybrid methods have been used in order to determine the best possible time-lag to represent the original time series [36]. A hybrid method that combined neural networks with a modified genetic algorithm was proposed to perform an evolutionary search for the minimum necessary time-lags for determining the phase space that generates the time series [37]. A morphological rank linear time-lag added evolutionary forecasting method was also proposed that carries out an evolutionary search for the lowest number of relevant time lags necessary to efficiently represent the patterns and characteristics of a complex time series [38]. A metaevolutionary algorithm simultaneously evolved both the neural networks and the set of time-series data that are needed to predict the time series [39]. The approach showed good results on a number of time-series problems where it was able to reconstruct the data set efficiently and accurately.

Multiobjective evolutionary algorithms have been used to optimize radial-basis networks for time-series prediction, which incorporated heuristics that were able to detect and remove networks which did not contribute much to the network output, while preserving those that produced good results [40]. The use of multiobjective evolutionary neural networks for time-series prediction employed training and validation accuracy as the two different objectives [34]. Multiple error measures have also been used as the different objectives in training evolutionary neural networks with multiobjective optimization [41]. Hybrid fuzzy model has been proposed for predicting nonlinear time-series data in which their two objectives were to improve prediction accuracy and minimize the number of required fuzzy rules [42]. A knee-point strategy multiobjective approach has shown promising results for evolving feedforward neural networks when compared with established multiobjective evolutionary algorithms [43]. Hybrid multiobjective evolutionary method has been used for evolution of recurrent neural network weights and structure with ensembles where a set of Pareto solutions are obtained [44] and has shown promising results.

B. Cooperative Coevolution for Neuroevolution

CC divides a large problem into smaller subcomponents, which are implemented as subpopulations that are evolved in isolation and cooperation takes place for fitness evaluation [1]. The subcomponents are also referred as modules. Problem decomposition determines how the problem is broken down into subcomponents. The size of a subcomponent and the way it is encoded depends on the problem. The original CC framework has been used for general function optimization 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 effective only for problems that are fully separable [2]. Much work has been done in the use of CC in large-scale function optimization, and the focus has been on nonseparable problems [2], [45]–[47].

A function of n variables is separable if it can be written as a sum of n functions with just one variable [48]. Nonseparable problems have interdependencies between variables as opposed to separable ones. Real-world problems mostly fall between fully separable and fully nonseparable. CC has been effective for separable problems. Evolutionary algorithms without any decomposition strategy appeal to fully nonseparable problems [4].

The subpopulations in CC 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 that have fewer interactions among themselves [4].

CC methods have been used for neuroevolution of recurrent neural networks for time-series problems [11], [17], and it has been shown that they perform better when compared with several methods from literature.

C. Diversity in Cooperative Coevolution

Population diversity is a key issue in the performance of evolutionary algorithms. The diversity of a population affects the convergence of a evolutionary algorithm. A population, which consists of similar candidate solutions in the initial stages of the search, is prone to convergence in a local minimum. The selection pressure and recombination operations mainly affect the diversity of the population. Evolutionary operators, such as crossover and mutation, must ensure that the population is diverse enough in order to avoid local convergence. Diverse candidate solutions can ensure the algorithm to escape a local minimum. In evolutionary algorithms, diversity has been improved using techniques, such as: 1) complex population structures [49], [50]; 2) the use of specialized operators to control and assist the selection pressure [51]; 3) reintroduction of genetic materials in the population [52], [53]; and 4) diversity measures, such as the hamming distance [54], gene frequencies [70], and diversity measures to explore and exploit search [55].

CC naturally retains diversity through the use of subpopulations, where mating is restricted to the subpopulations and cooperation, is mainly by collaborative fitness evaluation [1], [56]. Since selection and recombination are restricted to a subpopulation, the new solution will not have features from the rest of the subpopulations; therefore, CC produces more diverse population when compared with a standard evolutionary algorithm with a single population.

D. Recurrent Neural Networks for Time-Series Prediction

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 [18] 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 are given by

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

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.

In order to use neural networks for time-series prediction, the time-series data need to be preprocessed and reconstructed into a state space vector [57]. Given an observed time series $x(t)$, an embedded phase space $Y(t) = [(x(t), x(t-T), \ldots, x(t(D-1)T)]$ can be generated, where T is the time delay, D is the embedding dimension, $t = 0, 1, 2, \ldots, N - DT - 1$, and N is the length of the original time series [57]. 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 [58]. Taken's proved that if the original attractor is of dimension d, then $D = 2d + 1$ will be sufficient to reconstruct the attractor [57].

The reconstructed vector is used to train the recurrent network for one-step-ahead prediction where one 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 [11], [28], [30].

Either the root-mean-squared error (RMSE) or the normalized mean-squared error (NMSE) can be used to measure the prediction performance of the given recurrent neural network.

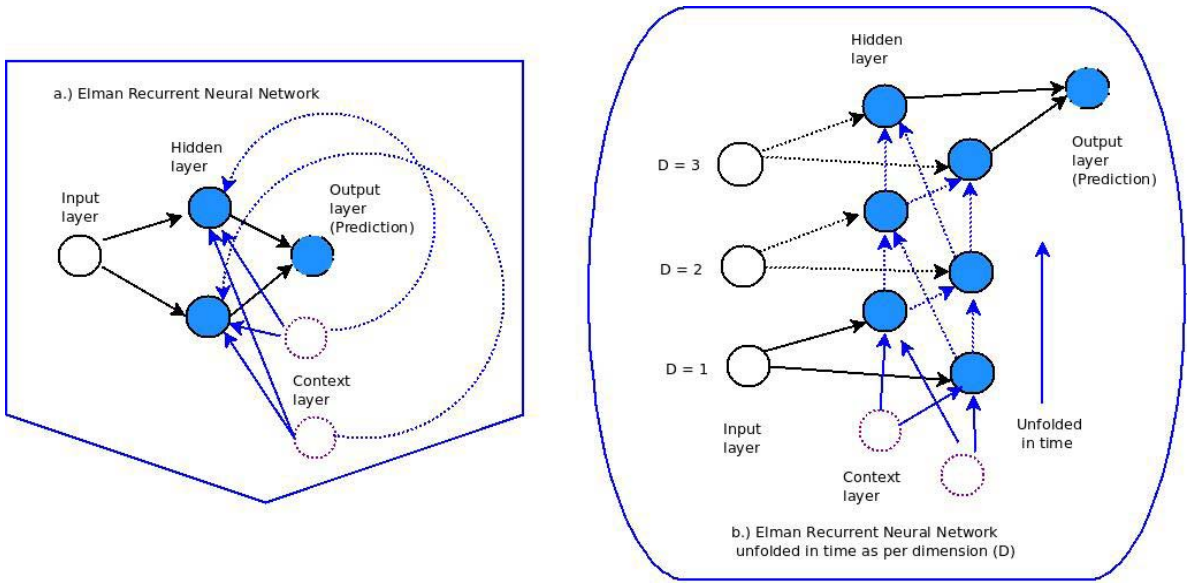


Fig. 1. Elman recurrent neural network used for time-series prediction. Note that only one neuron is used in the input and output layer. The number of hidden neurons varies as per application. The network unfolds in time according to the size of the dimension (D) from using Taken's theorem in obtaining state space vector from the time series. Solid lines (synapses): trainable weights that are evolved using the proposed competitive coevolution algorithm.

These are given in

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

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

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.

Elman recurrent neural networks given in (1) are shown in Fig. 1, where the network is given along with how it is unfolded through time according to the dimension (D). This approach has been used in our previous work [11]. We use a fixed dimension size; however, the architecture can also use the dimension size that varies for different points in the time series.

E. Problem Decomposition for Recurrent Networks

Problem decomposition is an important procedure in using CC for neuroevolution. The problem decomposition method will determine which set of weights from the neural network will be encoded into a particular subpopulation of CC. 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 neuroevolution that decompose the network on the NL and SL. In SL problem decomposition, the neural network is decomposed to its lowest level, where each weight connection (synapse) forms a subcomponent. Examples include cooperatively coevolved synapse neuroevolution [5] and neural fuzzy network with cultural cooperative particle swarm optimization (CPSO) [10]. In NL problem decomposition, the neurons in the network act as the reference point for the decomposition. Examples include enforced subpopulations [8], [9] and neuron-based subpopulation [6], [7].

III. Competition and Collaboration in Cooperative Coevolution

Collaboration 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 CC, the species are implemented as subpopulations that do not exchange genetic material with other subpopulations. Collaborations and exchange of genetic material or information between the subpopulations can be helpful in the evolutionary process. Competition and collaboration are vital component of evolution where different groups of species compete for resources in the same environment. Different types of problem decomposition methods in CC represent different groups of species (neural and SL [5]–[7]) in an environment that features collaboration through fitness evaluation during evolution.

In this section, we propose a CC 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 that reflects on the different degree of nonseparability (interaction of variables) and diversity (number of subpopulations) during evolution [4].

The proposed method is called competitive island-based CC (CICC), which employs different problem decomposition methods that compete with different features they have in terms of diversity and degree of nonseparability. In the rest

Algorithm 1 Competitive Two-Island CC for Training Recurrent Neural Networks

Stage 1: Initialisation:

i. Cooperatively evaluate Neuron level
ii. Evaluate Network level

Stage 2: Evolution:

```
while FuncEval ≤ GlobalEvolutionTime do
    while FuncEval ≤ 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 ≤ 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 3: Competition: Compare and mark the island with best
    fitness.

    Stage 4: Collaboration: Inject the best individual from the island
    with better fitness into the other island.
    if SL≤ NL then
        Copy NL best into chosen SL Individual.
    end
    else
        Copy SL best into chosen NL Individual.
    end
end
```

Algorithm 2 Competitive Three-Island CC for Recurrent Networks

Stage 1: Initialisation:

i. Cooperatively evaluate Synapse
ii. Cooperatively evaluate Neural level
iii. Evaluate Network level

Stage 2: Evolution:

```
while FuncEval ≤ GlobalEvolutionTime do
    while FuncEval ≤ 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 ≤ 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
    while FuncEval ≤ Island-Evolution-Time do
        while n Generations do
            Create new individuals using genetic operators
        end
    end
    Stage 3: Competition

    Stage 4: Collaboration: Inject the best individual from the island
    with better fitness into the other islands.
    if (SL≤ NL ) and (SL≤ NetL) then
        i. Copy SL best into chosen NL Individual
        ii. Copy SL best into chosen NetL Individual
    end
    else if (NL ≤ SL) and (NL≤ NetL) then
        i. Copy NL best into chosen SL Individual
        ii. Copy NL best into chosen NetL Individual
    end
    else
        i. Copy NetL best into chosen NL Individual
        ii. Copy NetL best into chosen SL Individual
    end
end
```

of the discussion, we refer to the different types of problem decomposition as islands. The proposed method features competition where 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 neuroevolution, only two or three 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 as follows.

1) *SL 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) *NL Problem Decomposition:* Decomposes the network into NL. The number of neurons in the hidden, state, and output layer determines the number of subcomponents [7].
3) *NetL:* The standard neuroevolution where only one population represents the entire network. There is no decomposition present at this level of encoding.

The proposed CICC methods for two and three islands are given in Algorithms 1 and 2, respectively. In Algorithm 1, initially, all the subpopulations of the SL and NL islands shown in Fig. 2 are initialized and evaluated using the framework shown in Fig. 4. In Stage 1, the subpopulations at SL and NL 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, which is given by the number of cycles that makes the required number of function evaluations in the respective islands. A cycle in CC is when all the subpopulations have been evolved for n number of generations in a round-robin fashion.

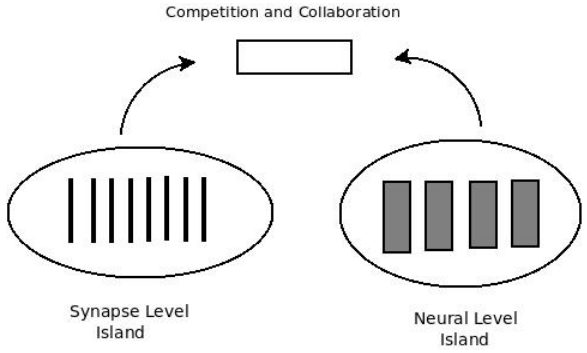


Fig. 2. Two-island CICC method that employs NL and SL islands.

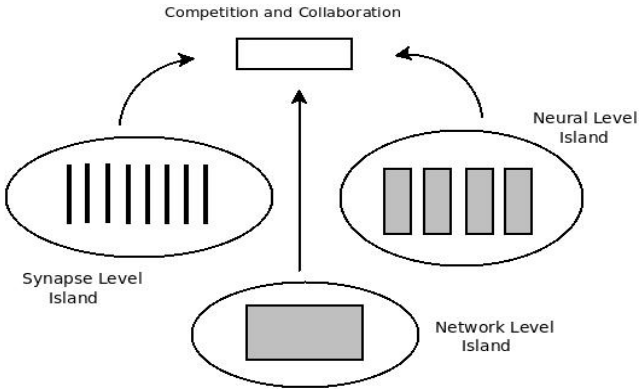


Fig. 3. Three-island CICC method that employs neural level, SL, and NetL islands that compete and collaborate by sharing the best solutions during evolution.

Once a particular island has been evolved for the island evolution time, the algorithm proceeds and checks if the best solution of the particular island is better than that of 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. In this way, the best solution is used to help the rest of the islands. Afterward, when the particular island changes, the best solution competes within the rest of the solutions from the same island until the local evolution time has been reached. In the collaboration procedure, the algorithm needs to consider how the solution from one island will be transferred to the rest of the islands.

In Algorithm 2, the three-island method follows the same approach as Algorithm 1, the difference being that there is an additional island (NetL) in this method, as shown in Fig. 3. The algorithm initializes and evaluates the respective islands before evolution begins in Stage 2, where all of the islands are evolved for the specified island evolution time. The island with the best fitness is then marked as shown in Stage 3. Stage 4 of both algorithms describes how the collaboration feature transfers the strongest individuals from the best island into the rest of the islands.

A. Cooperative Evaluation

Cooperative evaluation of individuals in the respective subpopulations is done by concatenating the chosen individual from a given subpopulation with the best individuals from the rest of the subpopulations [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.

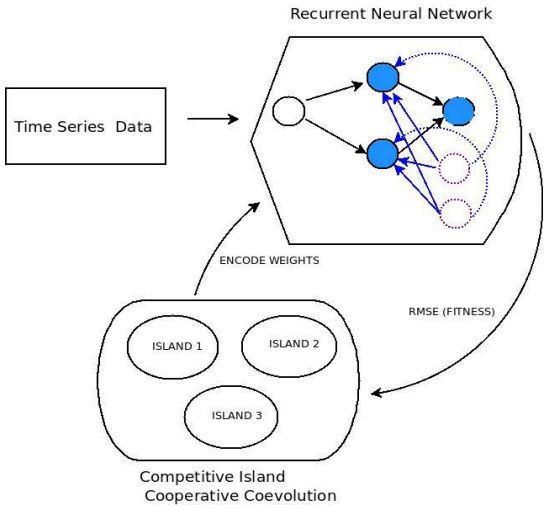


Fig. 4. Overall framework. The time-series data are reconstructed into state space vector using Taken's theorem. Each of the islands encodes the weights into the recurrent neural network in order to obtain the fitness given by the RMSE.

B. Competition

Each island employs a different problem decomposition method. In the SL island, a much higher number of function evaluation is required for a single cycle when compared with the NL island. The number of function evaluation depends on the number of subpopulations used in the island. SL island employs the highest number of subpopulations, as each weight link is represented as a subpopulation, whereas NL subpopulations have more than one weight variables.

The respective islands need to be given the same time for evolution; therefore, the number of function evaluations required needs to be the same or similar. We can only evolve the particular 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 better solution is then injected (copied) into the other islands, as shown in Stage 4 of Algorithms 1 and 2. A number of factors need to be considered when making 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 need to be carefully concatenated into an individual and transferred without losing any genotype (subcomponents in CC) to phenotype (recurrent neural network) mapping.

The winner island is used to inject the best solution to the other island. The island in which the best individual is

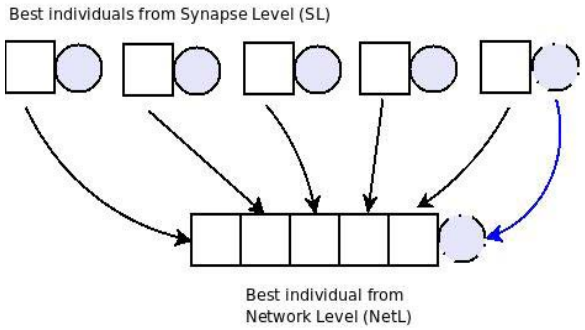


Fig. 5. Individuals shown as square box is copied from SL island into NetL island. A single fitness from the individuals (shown as circles) is copied. Note that only the fitness of the last individual is copied from SL island to NetL island. This fitness is the main fitness of the SL island.

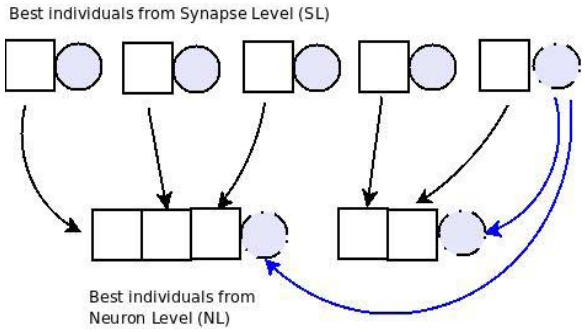


Fig. 6. Individuals shown as square box are copied from the SL island into the NL island. A single fitness from the individuals shown as circles is copied. Note that only the fitness of the last individual is copied from the SL island to the NL. This fitness is the main fitness of the SL island.

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 subpopulations 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. The evaluation depends on the type of the evolutionary algorithm used in the subpopulation. The fitness of the injected individual and the best individual in the subpopulation needs to be marked. Evaluation of the entire subpopulation is costly in terms of function evaluations.

Since each subpopulation contains individuals that have fitness, we need to note that there will be a number of different fitness values from the best individual in each subpopulation. We only take the best fitness value and use it to replace the best individuals from all the subpopulations in the other islands, as shown in Figs. 5–7. Since the number of subpopulations is different, only the best fitness replaces the old best fitness, as it carries a stronger solution.

In Fig. 5, best individuals from the SL island are transferred to NetL. Similar approach will be used in the reverse case where NetL is transferred to SL; however, the NetL fitness will replace each of the best individuals corresponding fitness in the SL. In Fig. 6, best individuals from SL island is transferred to NL. In the reverse case, the fitness of the last individual of the NetL will be copied to each of the best individuals of the SL. In Fig. 7, NL island is transferred to the NetL.

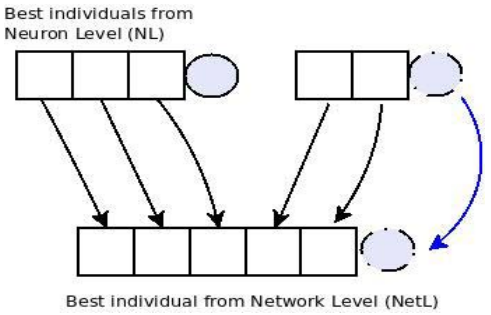


Fig. 7. Individuals shown as square box are copied from the NL island into the NetL island. A single fitness from the individuals shown as circles is copied.

The same trend as in the previous cases will be applied for reverse transfer.

D. Evolution Algorithm in the Subpopulations

The type of evolutionary algorithm used in the subpopulation will have certain requirements for such a transfer of solution to take place. In our implementation, we used the generalized generation gap with parent-centric crossover (G3-PCX) evolutionary algorithm [59] in the subpopulations.

The details of the G3-PCX are given as follows. The generalized generation gap differs from a standard genetic algorithm in terms of selection and creation of new individuals. In G3-PCX, the whole population is randomly initialized and evaluated similarly to the standard genetic algorithm. The difference lies in the optimization phase, where a small subpopulation is chosen. At each generation, n best fit and m random individuals are chosen from the main population to make up a subpopulation. The subpopulation is evaluated at each generation, and the evaluated individuals are added to the main population. In this way, over time, the individuals of the main populations are evaluated.

The best individual in the population is retained at each generation. The parent-centric crossover operator is used in creating an offspring based on orthogonal distance between the parents [59]. The parents are made of female and male components. The offspring is created in the neighborhood of the female parent. The male parent defines the range of the neighborhood. The neighborhood is the distance of the search space from the female parent, which is used to create the offspring. The genes of the offspring extract values from intervals associated with the neighborhood of the female and the male using a probability distribution. The range of this probability distribution depends on the distances among the genes of the male and the female parent. The parent-centric crossover operator assigns more probability to create the offspring near the female than anywhere else in the search space.

E. Diversity Through Competition and Collaboration

CC naturally retains diversity through the use of subpopulations, where mating is restricted to the subpopulations and cooperation is mainly by collaborative fitness evaluation [1], [56].

The proposed method employs competition through the islands, and the collaborative features ensure that the diversity of the islands is improved. Each island ensures a certain level of diversity due to the different problem decomposition methods and the number of subpopulations. Diversity can help in escaping from a local minimum through interisland competition and collaboration.

IV. Simulation and Analysis

This section presents an experimental study of CICC for training recurrent neural networks on chaotic time-series problems. The NL [11] and 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 [60] and Lorenz time series [20] are the two simulated time series, while the real-world problems are the Sunspot time series [61] and the financial time series from ACI Worldwide Inc. given in NASDAQ stock exchange [62].

The behavior of the respective methods is evaluated on different recurrent network topologies that are given by different numbers of hidden neurons. The size and description of the respective data set are taken from our previous work for a fair comparison [11]. The results are further compared with that of other computational intelligence methods in the literature.

A. Problem Description

The Mackay-Glass time series has been used in the literature as a benchmark problem due to its chaotic nature [60]. The differential equation used to generate the Mackey-Glass time series is given in

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

In (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 [29], [31], [63], and [64], where the constants $a = 0.2$, $b = 0.1$, and $c = 10$. The chaotic time series is generated using time delay $\tau = 17$ and initial value $x(0) = 1.2$.

The experiments use the chaotic time series with the length of 1000 generated by (4). The first 500 samples are used for training the Elman network, whereas the 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 Lorenz who has extensively contributed to the establishment of Chaos theory [20]. The Lorenz set of equations is given in (5), where σ, r, and b are dimensionless parameters

$$\begin{aligned} \frac{dx(t)}{dt} &= \sigma[y(t) - x(t)] \\ \frac{dy(t)}{dt} &= x(t)[r - z(t)] - y(t) \\ \frac{dz(t)}{dt} &= x(t)y(t) - bz(t). \end{aligned} \tag{5}$$

The typical values of these parameters are $\sigma = 10$, $r = 28$, and $b = 8/3$ [29], [31], [65]–[67]. The x-coordinate of the Lorenz time series is chosen for prediction and 1000 samples are generated. The time series is scaled in the range $[-1, 1]$. The first 500 samples are used for training and the remaining 500 is used for testing. The phase space of the original time series is reconstructed with the embedding dimensions $D = 3$ and $T = 2$.

The Sunspot time series is a good indication of the solar activities for solar cycles, which impacts the earth's climate, weather patterns, satellite, and space missions [68]. 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 [61]. 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 of the proposed methods with those of the methods in [29] and [31]. The time series is scaled in the range $[-1, 1]$. The first 500 samples are used for training, and the remaining 500 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 $[-1, 1]$ is done as in the literature in order to provide a fair comparison.

The financial time series data set is taken from the NASDAQ stock exchange [62]. It contains daily closing prices of 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 ~800 data points. We used embedding dimension $D = 5$ and time $T = 2$ to reconstruct the time-series data using Taken's theorem in order to get the training and testing data sets and obtain 200 for each set. The closing stock prices were normalized between 0 and 1. The data set also overlaps with the recession that hit the U.S. market. The given data points were divided into training and testing using a 50–50 split.

B. Experimental Setup

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 experimental setup is the same as our previous works [11]. The RMSE and NMSE given in (2) and (3) are used as the main performance measures of the recurrent network.

In the proposed CICC shown in Algorithms 1 and 2, each subpopulation is evolved for a fixed number of generations in a round-robin fashion. This is considered as the depth of search. Our previous work has shown that the depth of search of one generation gives optimal performance for both NL and SL decomposition [7]. Hence, one is used as the depth of search in all the experiments. Note that all subpopulations evolve for the same depth of search.

The termination condition of the all the problems and recurrent network training methods is when a total

TABLE I
TRAINING AND GENERALIZATION PREDICTION PERFORMANCE FOR THE MACKEY-GLASS TIME SERIES GIVEN IN (×E-02)

Method	H	Training	General.	Best
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
NetL	3	0.894 ± 0.073	0.898 ±0.074	0.504
	5	0.874 ± 0.065	0.878 ± 0.066	0.577
	7	1.160± 0.087	1.172 ± 0.089	0.860
	9	1.897±0.239	1.904 ± 0.239	1.089
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
Two-Island)	7	0.846 ±0.063	0.847 ± 0.064	0.470
	9	0.856 ±0.074	0.858 ±0.074	0.400
CICC	3	0.786 ± 0.0528	0.789 ± 0.0536	0.327
(SL-NL-NetL	5	0.791± 0.0453	0.792± 0.0458	0.520
Three-Island)	7	1.076 ± 0.0591	1.081 ± 0.059	0.589
	9	1.210 ± 0.101	1.216 ± 0.102	0.574

TABLE II
TRAINING AND GENERALIZATION PREDICTION PERFORMANCE FOR THE LORENZ TIME SERIES GIVEN IN (×E-02)

PD	H	Training	General.	Best
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
NetL	3	1.215 ± 0.172	1.244 ± 0.178	0.468
	5	1.102 ± 0.088	1.127 ± 0.091	0.516
	7	1.831 ± 0.177	1.896 ± 0.183	0.771
	9	2.613± 0.242	2.695 ± 0.248	1.360
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
Two-Island)	7	0.938 ±0.150	0.965 ±0.149	0.372
	9	0.888 ±0.097	0.915 ±0.101	0.442
CICC	3	1.739± 0.186	1.785 ± 0.194	0.567
(SL-NL-NetL	5	1.707 ± 0.174	1.779±0.182	0.527
Three-Island)	7	2.113 ± 0.193	2.201 ± 0.196	0.878
	9	2.352 ± 0.256	2.475 ± 0.267	0.864

TABLE III
TRAINING AND GENERALIZATION PREDICTION PERFORMANCE FOR THE SUNSPOT TIME SERIES GIVEN IN (×E-02)

PD	H	Training	General.	Best
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
NetL	3	1.485 ± 0.146	5.150 ± 1.868	1.579
	5	1.430 ± 0.102	4.283 ± 0.917	1.552
	7	1.924 ± 0.174	5.764 ± 1.564	1.663
	9	2.300 ± 0.124	8.801 ± 2.044	2.633
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
Two-Island)	7	1.348 ± 0.0745	7.606 ± 2.219	1.663
	9	1.485 ± 0.071	6.657 ± 1.771	1.778
CICC	3	1.830 ± 0.115	5.954 ± 1.629	1.902
(SL-NL-NetL	5	1.873 ± 0.113	6.509 ± 2.717	1.777
Three-Island)	7	2.090± 0.157	8.900 ± 2.975	2.347
	9	2.448 ± 0.157	10.646 ± 3.67163	3.429

TABLE IV
TRAINING AND GENERALIZATION PREDICTION PERFORMANCE FOR THE FINANCE (ACI WORLDWIDE INC.) TIME SERIES GIVEN IN (×E-02)

PD	H	Training	General.	Best
CC-NL	3	2.074 ± 0.041	2.117 ± 0.132	1.934
	5	2.027 ± 0.030	2.041 ± 0.024	1.931
	5	2.010 ± 0.019	2.043 ± 0.044	1.932
	5	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
	5	2.108 ± 0.051	2.108 ± 0.058	1.931
	5	2.106 ± 0.041	2.170 ± 0.065	1.947
NetL	3	2.001 ±0.023	2.070 ±0.142	1.927
	5	2.004 ±0.044	2.017 ±0.029	1.916
	7	2.054 ±0.027	2.031 ±0.029	1.910
	9	2.155 ±0.042	2.126 ±0.052	1.917
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
Two-Island)	7	1.941 ± 0.018	2.027 ± 0.030	1.935
	9	1.932 ± 0.019	2.005 ± 0.015	1.942
CICC	3	1.974 ± 0.011	2.043 ± 0.098	1.921
(SL-NL-NetL	5	1.989 ± 0.046	1.994± 0.012	1.927
Three-Island)	7	2.013 ± 0.020	2.014 ± 0.021	1.935
	9	2.070 ± 0.059	2.067 ± 0.050	1.934

of 50 000 function evaluations has been reached by the respective cooperative coevolutionary methods (CC-NL and CC-SL). The proposed CICC-two-island method employs a total of 100 000 function evaluation, where each island (SL and NL) employs 50 000 function evaluations. The proposed CICC-three-island method employs a total of 150 000 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 numbers of hidden neurons for Elman style recurrent networks using the respective coevolutionary algorithms given in Tables I–IV. The CC-NL and CC-SL represent standalone CC NL and SL methods, respectively. They are used to compare with the proposed CICC (SL-NL two island) and (SL-NL-NetL three island) methods using the same setup for the recurrent network architecture and optimization time in terms of function evaluations as given in Section IV-B.

The results report the RMSE with mean and 95% 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 show the best results. We first compare the results of the CICC-two-island method with standalone methods (NL and SL), as the proposed method involves the competition and collaboration between the two standalone methods. Later, we compare the three-island CICC method with the rest of the methods.

In Table I, the results of the Mackey-Glass time series show that the CICC-two-island method has given better performance than CC-NL and CC-SL. This is clear for all the cases, i.e., for three to nine hidden neurons. In Table II, similar trend is seen for the Lorenz problem, where CICC-two-island method outperforms standalone CC-SL and CC-NL. This is seen for the training, generalization, and the best runs. The performance has improved as the number of neurons increases, which indicates that CICC-two-island method scales better than SL and NL.

In the Mackey-Glass problem, we observe that the three-island approach shows the best performance for three and five neurons. It outperforms the two-island method and the rest of the standalone methods. The improvement in performance is due to the third island (NetL), which has been added to the competition and the collaborative features helped in improving the results. In the case of seven and nine neurons, it deteriorates in performance when compared with two-island method, but performs better than standalone methods.

In the Lorenz time series, the three-island method is not able to outperform the two-island method and the standalone methods. It also shows that the three-island method does not scale as well as the two-island and rest of the methods. The collaboration of the NetL island seems to influence and hence deteriorate its performance, as the size of the problem in terms of hidden neurons increases. The third island (NetL) has deteriorated the performance by injecting its best solution to the rest of the islands.

We note that both of these problems are stimulated time series that do not contain noise; hence, there was no problem faced in overfitting that is common for poor generalization performance.

The results in Tables III and IV reveal the performance of the proposed method for real-world time series where noise is present, and therefore we only consider the training performance, as the generalization performance is also dependent on overfitting during training. In the Sunspot time series, CICC-two-island method performs better than the other methods (NL and SL) for all the cases. It also scales better as the number of hidden neurons increases. The three-island method shows better performance than the standalone methods only for the case of three hidden neurons. The three-island method did not outperform the two-island approach. The three-island method seems to have deteriorated in the performance given by the two-island method by collaboration from the NetL island.

The same trend is seen for the finance time series (ACI Worldwide Inc.) problem. The two-island method has been able to outperform the standalone method (SL and NL). The three-island method has performed better than all the methods for three neurons only. The performance deteriorates as the number of hidden neurons increased due to collaboration from the NetL island.

The results show that the generalization performance is dependent on the neural network topology according to the number of hidden neurons. In the Sunspot time series, we observe that the training performance improves, as the number of hidden neurons increases; however, the generalization performance deteriorates possibly due to overfitting. The financial time series does not show major difference of the generalization performance. Although these problems are both real-world time series, both are from different domains and have different properties in the time series and, therefore, the performance has shown to be different.

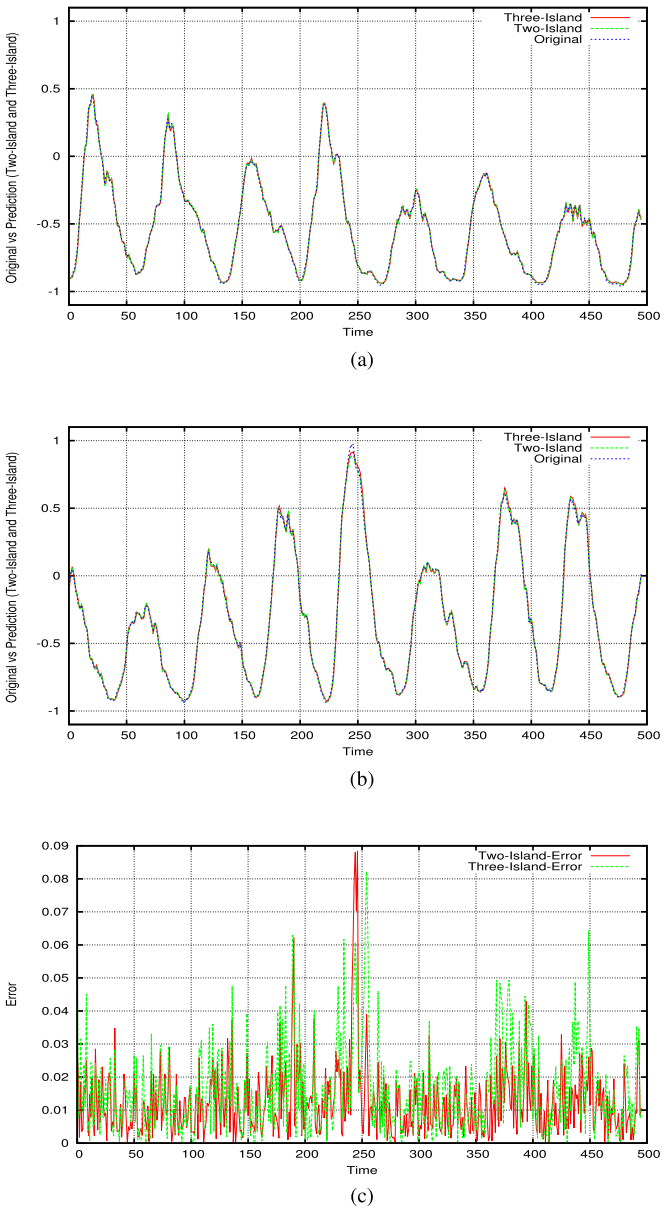
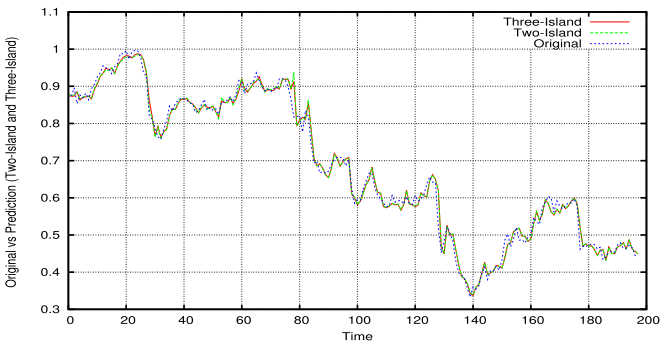


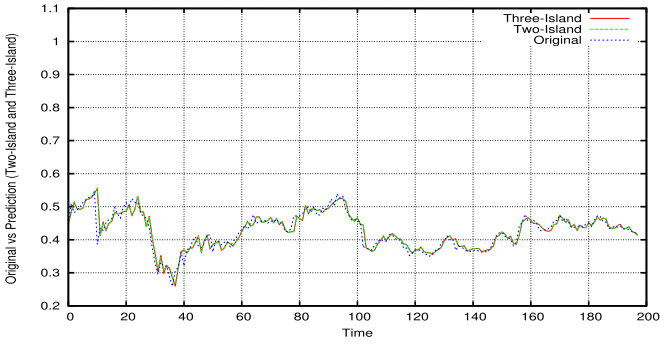
Fig. 8. Typical prediction given by two- and three-island CICC methods for Sunspot time series. The RMSE for the two-island method on the data is 1.572E-02. The RMSE for the three-island method on the test data is 1.777E-02. (a) Performance on the training data set. (b) Performance on the test data set. (c) Error on the test data set.

Figs. 8 and 9 show the best experimental run with the training and test prediction performance of the Sunspot and finance time series, respectively. The two-island approach and three-island approach show competitive performance, which is given by their error plot from the test prediction.

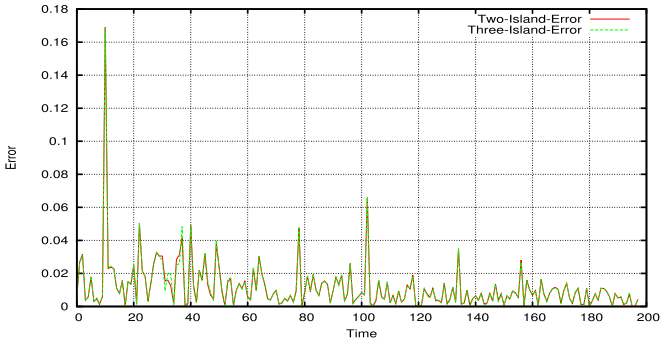
Tables V, VI and VII compare the best results from the previous tables with some of the established methods in the literature. The RMSE from the best experimental is used to compare



(a)



(b)



(c)

Fig. 9. Typical prediction given by two- and three-island CICC methods for Finance-ACI time series. The RMSE for the two-island method on the data is 1.920E-02. The RMSE for the three-island method on the test data is 1.921E-02. (a) Performance on the training data set. (b) Performance on the test data set. (c) Error on the test data set.

TABLE V
COMPARISON WITH THE RESULTS FROM THE LITERATURE ON THE LORENZ TIME SERIES

Prediction Method	RMSE	NMSE
Backpropagation-through-time (2010) [30]		1.85E-03
Real time recurrent learning (2010) [30]		1.72E-03
Recursive Bayesian LM Alg. (2010) [30]		9.0E-04
NARX-Elman-Residual Analysis (2010) [31]	1.08E-04	1.98E-10
BP-NN residual analysis (2011) [32]	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 [17]	5.06E-03	4.88E-04
CICC (Two Island) -RNN	3.55E-03	2.41E-04
CICC (Three Island) -RNN	5.27E-03	7.45E-05

along with the NMSE that was obtained particularly for comparison of results with literature. In the literature, in some cases, the mean result is given, which can be compared with

TABLE VI
COMPARISON WITH THE RESULTS FROM THE LITERATURE ON THE MACKEY-GLASS TIME SERIES

Prediction Method	RMSE	NMSE
Neural fuzzy network and PS0 (2009) [10]	2.10E-02	
Neural fuzzy - cooperative PS0 (2009) [10]	1.76E-02	
Neural fuzzy network and DE (2009) [10]	1.62E-02	
Neural fuzzy network and GA (2009)[10]	1.63E-02	
BPNN GA Residual Analysis (2011) [32]	1.30E-03	
NARX-Elman - Residual Analysis (2010) [31]	3.72E-05	2.70E-08
CCRNN-Synapse Level (2012) [11]	6.33E-03	2.79E-04
CCRNN-Neuron Level (2012) [11]	8.28E-03	4.77E-04
AMCC-RNN [17]	7.53E-03	3.90E-04
Type-2 Fuzzy Neural Networks [33]	3.90E-02	
Multi-objective RNN Ensembles [44]	7.53E-03	1.11E-03
CICC (Two Island) -RNN	4.00E-03	1.11E-04
CICC (Three Island) -RNN	3.27E-03	5.28E-04

TABLE VII
COMPARISON WITH THE RESULTS FROM THE LITERATURE ON THE SUNSPOT TIME SERIES

Prediction Method	RMSE	NMSE
Multi-layer perceptron (1996) [28]		9.79E-02
Elman RNN (1996) [28]		9.79E-02
FIR Network (MLP) (1996) [28]		2.57E-01
Wavelet packet MLP (2001)[69]		1.25E-01
Radial basis network (RBF-OLS)(2006) [29]		4.60E-02
Locally linear neuro-fuzzy (2006) [29]		3.20E-02
NARX-Elman -Residual Analysis (2010) [31]	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 [17]	2.41E-02	3.11E-03
Multi-objective RNN Ensembles [44]	1.56E-02	1.24E-03
CICC (Two Island) -RNN	1.34E-02	1.31E-03
CICC (Three Island) -RNN	1.77E-02	1.67E-03

the results given in Tables I, II and III, respectively. We are interested in comparison of the results with our previous works as they have used the same data and experimental setup and, therefore, a fair comparison can be done with them [11], [17]. We note particular financial time-series data set was used an application, and we did not find any work done in the literature for comparison.

D. Discussion

The proposed CICC methods have given better performance when compared with similar evolutionary approaches, such as training neural fuzzy networks with hybrid of CCPSO, CPSO, genetic algorithms, and differential evolution [10]. The only exception is being the results from Hybrid NARX–Elman networks [31] as it has additional enhancements, such as the optimization of the embedding dimensions and strength of architectural properties of hybrid neural networks with residual analysis [31].

The results have also been compared with our past work, where CC of recurrent neural networks was used for the first time for time-series prediction. SL and NL problem decomposition methods were used and compared, and it was shown that the NL gave better performance for two out of the three problems [11]. Adaptation of problem decomposition method during evolution was done for recurrent neural networks for grammatical systems problems [16]. We applied the same method for chaotic time-series prediction [17] and got further improvements of the results when compared with our previous

work [11]; however, the adaptive problem decomposition method has limitations due to parameter settings, which makes it time-consuming. The adaptive problem decomposition was based on the experimental results given in our earlier works [4] that showed that the interdependence between the variables changes over time.

CICC-two- and three-island methods perform better than standalone CC in literature for Lorenz and Mackey-Glass problems problems using cooperative coevolutionary recurrent neural networks (CCRNN-SL and CCRNN-NL). In the Sunspot problem, the two-island method performed better than previous methods; however, the three island method did not outperform the CCRNN-SL, as shown in Table VII. The proposed methods perform better when compared with adaptive modularity CC where the motivation was to change the problem decomposition method with time, i.e., begin with SL and then move to NL and NetL, 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 determine when to change from one problem decomposition to another level and there is no established measure of the interacting variables as given by the degree of nonseparability [4].

SL island would be most useful in separable problems that have lower degree of nonseparability—it provides more flexibility and enforces global search through the subpopulations. CICC fulfills the limitations faced by fixed problem decomposition methods 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.

The test of scalability in the experiments has been observed through the behavior of the algorithms when the problem size in terms of hidden neurons increases. The two-island method has shown to have properties that give high level of scalability when compared with SL and NL standalone methods. The three-island method has shown to deteriorate in performance for larger number of hidden neurons in the case of both the simulated time-series problems. This can be due to the collaborative features where the best solution is shared with the rest of the islands. Due to several islands and competition, elitism is not fully ensured. This can be improved by have a separate population that keeps in track of the best results and provides elitism.

We have used two- and three-island methods that employed established problem decomposition methods. CICC can be further improved by different type of islands—which will depend on different problem decomposition methods. We need to replace the NetL island with an island that uses new type of problem decomposition as the NetL island has shown to deteriorate performance of the three-island method. The new island can be composed by problem decomposition where the number of subpopulation and its composition can be determined or chosen arbitrarily.

The solutions in each island are evolved with a fixed degree of nonseparability, which remains the same when a solution is injected or taken from another island. The quality of the solution helps other solutions within the island, as it competes and also shares its genetic material through operators, such as selection and crossover within the island. The transfer of the solution from one island to another also affects the diversity. Theoretical and experimental studies on how the degree of nonseparability is affected or how the global-local search benefits from the injection of solution can be done in future studies.

A major advantage of the proposed method is that it can be implemented in a multithreaded environment that will speed up the computation time that is a limitation of CC for training neural network when compared with gradient-based methods. In a multithreaded implementation, each island can run on a separate thread. In a multithreaded implementation, the overall training time can be lowered, which is a major limitation of the proposed competitive island-based method where the training time will increase as the number of islands increases.

V. Conclusion

Competition and collaboration are vital components in natural evolution. This paper presented CICC of recurrent neural networks for chaotic time-series prediction. The proposed approach employed two- and three-island competitive methods that were defined by different problem decomposition methods. The results have shown that the two-island method outperforms the standalone CC methods in terms of prediction performance and scalability. The three-island method has shown to perform better in few cases. The proposed methods perform better than several other methods from the literature. The proposed method takes advantage of problem decomposition methods with different degree of nonseparability and diversity. In a conventional CC 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 the 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 that employs diverse solutions from the rest of the islands.

In future work, the proposed method can be improved by exploring other problem decomposition methods that can provide more competition. A study of how the degree of nonseparability and its relationship to global–local search is affected when solutions are injected via the collaborate platform of the island can also be explored. A multithreaded 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 also be used for large-scale global optimization problems. Convergence proof of the CC-based training of recurrent networks can also be explored in future work.

References

[1] M. A. Potter and K. A. De Jong, "A cooperative coevolutionary approach to function optimization," in *Parallel Problem Solving from Nature—PPSN III* (Lecture Notes in Computer Science), vol. 866, Y. Davidor, H.-P. Schwefel, and R. Männer, Eds. Berlin, Germany: 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 *Proc. Congr. Evol. Comput.*, San Diego, CA, USA, Jun. 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, Jun. 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, Jun. 2008.

[6] R. Chandra, M. Frean, and M. Zhang, "An encoding scheme for cooperative coevolutionary feedforward neural networks," in *Proc. 23rd Austral. Joint Conf. Artif. Intell.*, 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. Miikkulainen, "Incremental evolution of complex general behavior," *Adapt. Behavior*, vol. 5, nos. 3–4, pp. 317–342, 1997.

[9] F. J. Gomez, "Robust non-linear control through neuroevolution," Dept. Comput. Sci., Univ. Texas Austin, Austin, TX, USA, Tech. Rep. 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," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 39, no. 1, pp. 55–68, Jan. 2009.

[11] R. Chandra and M. Zhang, "Cooperative coevolution of Elman recurrent neural networks for chaotic time series prediction," *Neurocomputing*, vol. 86, pp. 116–123, Jun. 2012.

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

[13] C. K. Goh, K. C. Tan, D. S. Liu, and S. C. Chiam, "A competitive and cooperative co-evolutionary approach to multi-objective particle swarm optimization algorithm design," *Eur. J. Oper. Res.*, 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," *IEEE Trans. Evol. Comput.*, 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 *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, San Jose, CA, USA, Jul. 2011, pp. 681–688.

[16] R. Chandra, M. Frean, and M. Zhang, "Adapting modularity during learning in cooperative co-evolutionary recurrent neural networks," *Soft Comput.-Fusion Found., Methodol., Appl.*, vol. 16, no. 6, pp. 1009–1020, 2012.

[17] R. Chandra, "Adaptive problem decomposition in cooperative coevolution of recurrent networks for time series prediction," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Dallas, TX, USA, Aug. 2013, pp. 1–8.

[18] J. L. Elman, "Finding structure in time," *Cognit. Sci.*, vol. 14, no. 2, pp. 179–211, 1990.

[19] R. Chandra, "Competitive two-island cooperative coevolution for training Elman recurrent networks for time series prediction," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Beijing, China, Jul. 2014, pp. 565–572.

[20] E. N. Lorenz, "Deterministic nonperiodic flow," *J. Atmos. Sci.*, vol. 20, no. 2, pp. 130–141, 1963.

[21] S. H. Kellert, *In the Wake of Chaos: Unpredictable Order in Dynamical Systems*. Chicago, IL, USA: Univ. Chicago Press, 1993.

[22] E. N. Lorenz, *The Essence of Chaos*. Seattle, WA, USA: Univ. Washington Press, 1993.

[23] H. Jiang and W. He, "Grey relational grade in local support vector regression for financial time series prediction," *Expert Syst. Appl.*, vol. 39, no. 3, pp. 2256–2262, 2012.

[24] B. Wang, H. Huang, and X. Wang, "A novel text mining approach to financial time series forecasting," *Neurocomputing*, vol. 83, pp. 136–145, Apr. 2012.

[25] H.-M. Feng and H.-C. Chou, "Evolutional RBFNs prediction systems generation in the applications of financial time series data," *Expert Syst. Appl.*, vol. 38, no. 7, pp. 8285–8292, 2011.

[26] X. Liang, R.-C. 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, Sep. 2013.

[27] A. Azzini, C. da Costa Pereira, and A. G. B. Tettamanzi, "Predicting turning points in financial markets with fuzzy-evolutionary and neuro-evolutionary modeling," in *Applications of Evolutionary Computing* (Lecture Notes in Computer Science), vol. 5484, M. Giacobini *et al.*, Eds. Berlin, Germany: Springer-Verlag, 2009, pp. 213–222.

[28] T. Koskela, M. Lehtokangas, J. Saarinen, and K. Kaski, "Time series prediction with multilayer perceptron, FIR and Elman neural networks," in *Proc. World Congr. Neural Netw.*, San Diego, CA, USA, 1996, pp. 491–496.

[29] 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, no. 3, pp. 217–239, 2006.

[30] D. T. Mirikitani and N. Nikolaev, "Recursive Bayesian recurrent neural networks for time-series modeling," *IEEE Trans. Neural Netw.*, vol. 21, no. 2, pp. 262–274, Feb. 2010.

[31] M. Ardalani-Farsa and S. Zolfaghari, "Chaotic time series prediction with residual analysis method using hybrid Elman–NARX neural networks," *Neurocomputing*, vol. 73, nos. 13–15, pp. 2540–2553, 2010.

[32] M. Ardalani-Farsa and S. Zolfaghari, "Residual analysis and combination of embedding theorem and artificial intelligence in chaotic time series forecasting," *Appl. Artif. Intell., Int. J.*, vol. 25, no. 1, pp. 45–73, 2011.

[33] F. Gaxiola, P. Melin, F. Valdez, and O. Castillo, "Interval type-2 fuzzy weight adjustment for backpropagation neural networks with application in time series prediction," *Inf. Sci.*, vol. 260, pp. 1–14, Mar. 2014.

[34] S. Chiam, K. Tan, and A. Mamun, "Multiobjective evolutionary neural networks for time series forecasting," in *Evolutionary Multi-Criterion Optimization* (Lecture Notes in Computer Science), vol. 4403, S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, Eds. Berlin, Germany: Springer, 2007, pp. 346–360.

[35] M. Ragulskis and K. Lukoseviciute, "Non-uniform attractor embedding for time series forecasting by fuzzy inference systems," *Neurocomputing*, vol. 72, nos. 10–12, pp. 2618–2626, 2009.

[36] R. de A. Araujo, A. L. I. de Oliveira, and S. C. B. Soares, "A quantum-inspired hybrid methodology for financial time series prediction," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Barcelona, Spain, Jul. 2010, pp. 1–8.

[37] T. Ferreira, G. C. Vasconcelos, and P. J. L. Adeodato, "A new evolutionary approach for time series forecasting," in *Proc. IEEE Symp. Comput. Intell. Data Mining*, Honolulu, HI, USA, Mar./Apr. 2007, pp. 616–623.

[38] R. de A. Araujo, R. L. Aranildo, and T. Ferreira, "Morphological-rank-linear time-lag added evolutionary forecasting method for financial time series forecasting," in *Proc. IEEE Congr. Evol. Comput.*, Hong Kong, Jun. 2008, pp. 1340–1347.

[39] E. Parras-Gutierrez and V. M. Rivas, "Time series forecasting: Automatic determination of lags and radial basis neural networks for a changing horizon environment," in *Proc. Int. Joint Conf. Neural Netw.*, Barcelona, Spain, Jul. 2010, pp. 1–7.

[40] J. González, I. Rojas, H. Pomares, and J. Ortega, "RBF neural networks, multiobjective optimization and time series forecasting," in *Connectionist Models of Neurons, Learning Processes, and Artificial Intelligence* (Lecture Notes in Computer Science), vol. 2084, J. Mira and A. Prieto, Eds. Berlin, Germany: Springer-Verlag, 2001, pp. 498–505.

[41] J. E. Fieldsend and S. Singh, "Pareto evolutionary neural networks," *IEEE Trans. Neural Netw.*, vol. 16, no. 2, pp. 338–354, Mar. 2005.

[42] M. R. Hassan, B. Nath, M. Kirley, and J. Kamruzzaman, "A hybrid of multiobjective evolutionary algorithm and HMM-fuzzy model for time series prediction," *Neurocomputing*, vol. 81, pp. 1–11, Apr. 2012.

[43] W. Du, S. Y. S. Leung, and C. K. Kwong, "Time series forecasting by neural networks: A knee point-based multiobjective evolutionary algorithm approach," *Expert Syst. Appl.*, vol. 41, no. 18, pp. 8049–8061, 2014.

[44] C. Smith and Y. Jin, "Evolutionary multi-objective generation of recurrent neural network ensembles for time series prediction," *Neurocomputing*, vol. 143, pp. 302–311, Nov. 2014.

[45] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 225–239, Jun. 2004.

[46] Y.-J. Shi, H.-F. Teng, and Z.-Q. Li, "Cooperative co-evolutionary differential evolution for function optimization," in *Advances in Natural Computation* (Lecture Notes in Computer Science), vol. 3611, L. Wang, K. Chen, and Y. S. Ong, Eds. Berlin, Germany: Springer-Verlag, 2005, pp. 1080–1088.

[47] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Inf. Sci.*, vol. 178, no. 15, pp. 2985–2999, 2008.
[48] D. Ortiz-Boyer, C. Hervá-Martínez, and N. García-Pedrajas, "CIXL2: A crossover operator for evolutionary algorithms based on population features," *J. Artif. Intell. Res.*, vol. 24, no. 1, pp. 1–48, 2005.
[49] R. K. Ursem, "Multinational evolutionary algorithms," in *Proc. Congr. Evol. Comput.*, Trondheim, Norway, May 1999, pp. 1633–1640.
[50] R. Thomsen, P. Rickers, and T. Krink, "A religion-based spatial model for evolutionary algorithms," in *Proc. 6th Int. Conf. Parallel Problem Solving Nature*, 2000, pp. 817–826.
[51] D. E. Goldberg and J. Richardson, "Genetic algorithms with sharing for multimodal function optimization," in *Proc. 2nd Int. Conf. Genet. Algorithms Genet. Algorithms Appl.*, Hillsdale, NJ, USA, 1987, pp. 41–49.
[52] H. G. Cobb and J. J. Grefenstette, "Genetic algorithms for tracking changing environments," in *Proc. 5th Int. Conf. Genet. Algorithms*, 1993, pp. 523–530.
[53] G. W. Greewood, G. B. Fogel, and M. Ciobanu, "Emphasizing extinction in evolutionary programming," in *Proc. Congr. Evol. Comput.*, Trondheim, Norway, May 1999, pp. 666–671.
[54] H. Shimodaira, "A diversity control oriented genetic algorithm (DCGA): Development and experimental results," in *Proc. Genet. Evol. Comput. Conf.*, 1999, pp. 603–611.
[55] R. K. Ursem, "Diversity-guided evolutionary algorithms," in *Proc. 7th Int. Conf. Parallel Problem Solving Nature*, 2002, pp. 462–474.
[56] M. A. Potter and K. A. De Jong, "Cooperative coevolution: An architecture for evolving coadapted subcomponents," *Evol. Comput.*, vol. 8, no. 1, pp. 1–29, 2000.
[57] F. Takens, "Detecting strange attractors in turbulence," in *Dynamical Systems and Turbulence, Warwick 1980* (Lecture Notes in Mathematics). Berlin, Germany: Springer-Verlag, 1981, pp. 366–381.
[58] C. Frazier and K. M. Kockelman, "Chaos theory and transportation systems: Instructive example," *Transp. Res. Rec., J. Transp. Res. Board*, vol. 1897, no. 1, pp. 9–17, 2004.
[59] 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.
[60] M. C. Mackey and L. Glass, "Oscillation and chaos in physiological control systems," *Science*, vol. 197, no. 4300, pp. 287–289, 1977.
[61] SILSO World Data Center. *The International Sunspot Number (1834–2001), International Sunspot Number Monthly Bulletin and Online Catalogue*. Royal Observatory Belgium, Brussels, Belgium. [Online]. Available: http://www.sidc.be/silso/, accessed Feb. 2, 2015.
[62] *NASDAQ Exchange Daily: 1970–2010 Open, Close, High, Low and Volume*. [Online]. Available: http://www.nasdaq.com/symbol/aciw/stock-chart, accessed Feb. 2, 2015.
[63] J.-S. R. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, no. 3, pp. 665–685, May/Jun. 1993.
[64] I. Rojas *et al.*, "Time series analysis using normalized PG-RBF network with regression weights," *Neurocomputing*, vol. 42, nos. 1–4, pp. 267–285, 2002.
[65] M. Assaad, R. Boné, and H. Cardot, "Predicting chaotic time series by boosted recurrent neural networks," in *Neural Information Processing* (Lecture Notes in Computer Science), vol. 4233, I. King, J. Wang, L.-W. Chan, and D. Wang, Eds. Berlin, Germany: Springer-Verlag, 2006, pp. 831–840.
[66] Q.-L. Ma, Q.-L. Zheng, H. Peng, T.-W. Zhong, and L.-Q. Xu, "Chaotic time series prediction based on evolving recurrent neural networks," in *Proc. Int. Conf. Mach. Learn. Cybern.*, Hong Kong, Aug. 2007, pp. 3496–3500.
[67] I. Rojas *et al.*, "Soft-computing techniques and ARMA model for time series prediction," *Neurocomputing*, vol. 71, nos. 4–6, pp. 519–537, 2008.
[68] S. Sello, "Solar cycle forecasting: A nonlinear dynamics approach," *Astron. Astrophys.*, vol. 377, no. 1, pp. 312–320, 2001.
[69] K. K. Teo, L. Wang, and Z. Lin, "Wavelet packet multi-layer perceptron for chaotic time series prediction: Effects of weight initialization," in *Proc. Int. Conf. Comput. Sci.-II*, 2001, pp. 310–317.
[70] M. M. Gouvêa, Jr., and A. F.R. Araújo, "A population dynamics model to describe gene frequencies in evolutionary algorithms," *Appl. Soft Comput.*, vol. 12, no. 5, pp. 1483–1492, May 2012.

Rohitash Chandra received the B.S. degree in computer science and engineering technology from the University of the South Pacific, Suva, Fiji, the M.S. degree in computer science from the University of Fiji, Lautoka, Fiji, and the Ph.D. degree in computer science from the Victoria University of Wellington, Wellington, New Zealand.

He is currently a Lecturer of Computer Science with the School of Computing, Information and Mathematical Sciences, University of the South Pacific. His current research interests include methodologies and applications of artificial intelligence with an emphasis on neural and evolutionary computation. He is the Founder and President of Software Foundation, Nausori, Fiji, which is a nonprofit organization that promotes the development of software and technology in Fiji. Apart from his interest in science, he has been involved in literature and philosophy with an emphasis on poetry. His third poetry collection entitled *Being at Home* was published in 2014.