Memetic cooperative neuro-evolution for chaotic time series prediction

Gary Wong[1], Rohitash Chandra[2] and Anuraganand Sharma[1]

[1] School of Computing Information and Mathematical Sciences
University of the South Pacific, Suva, Fiji.
[2] Artificial Intelligence and Cybernetics Research Group
Software Foundation, Nausori, Fiji.
s11085179@student.usp.ac.fj, c.rohitash@gmail.com,
anuraganand.sharma@usp.ac.fj

Abstract. Cooperative neuro-evolution has shown to be promising for chaotic time series problem as it provides global search features using evolutionary algorithms. Back-propagation features gradient descent as a local search method that has the ability to give competing results. A synergy between the methods is needed in order to exploit their features and achieve better performance. Memetic algorithms incorporate local search methods for enhancing the balance between diversification and intensification. We present a memetic cooperative neuro-evolution method that features gradient descent for chaotic time series prediction. The results show that the proposed method utilizes lower computational costs while achieving higher prediction accuracy when compared to related methods. In comparison to related methods from the literature, the proposed method has favorable results for highly noisy and chaotic time series problems.

Keywords: Memetic algorithms, cooperative neuro-evolution, backpropagation, gradient descent, and feedforward networks

1 Introduction

Evolutionary Algorithms (EAs) have achieved significant success as search and optimization techniques across various domains [11, 8]. In particular, they are best suited for non-linear and noisy systems [4], and have gained success for training neural networks which are also known as neuro-evolution [25]. A major drawback of EAs is convergence due to computational costs as they are essentially black-box optimization methods. Gradient-based training methods provide much faster convergence which has been successful for convex optimization, howsoever, premature convergence and over-training have been a recurring challenge [2].

The balance between diversification and intensification for neuro-evolution is imperative in order to provide more emphasis on promising solutions in the search space [20, 17, 14]. Memetic algorithms (MAs) [17] take advantage of the strengths of evolutionary algorithms and local search methods while eliminating their weaknesses [20, 9, 4, 2, 13]. Successful implementations of MAs span over many fields including plane wing design [20], image classification [1], continuous function optimization [2], and pattern classification [9].

Cooperative coevolution decomposes a problem into subcomponents [21] which are typically implemented as sub-populations that evolve in isolation and cooperation takes place for fitness evaluation. Cooperative neuro-evolution (CNE) refers to the application of cooperative coevolution for training neural networks. CNE has been applied successfully for time series prediction [8]. A memetic CNE method was presented with crossover-based local search and was able to reduce the computational time while achieving high-quality solutions for pattern classification problems [9]. Moreover, it has been shown that gradient descent algorithms converge faster with high multi-modal problems that complement in developing memetic algorithms [14]. There has not been much work done that features local search in improving CNE for chaotic time series problems.

In this paper, we present memetic cooperative neuro-evolution method that features gradient descent for chaotic time series problems. We evaluate the performance of the approach on simulated and real-world benchmark chaotic time series and also compare the results with related methods from the literature.

The rest of the paper is organized as follows. Section 2 presents the proposed method and Section 3 focuses on the experiment design and results. Consequently, Section 4 presents conclusion and discussion for future work.

2 Memetic Cooperative Neuro-evolution with Gradient Local Search

As discussed earlier, Chandra et. al presented a framework for memetic cooperative coevolution for pattern classification [9] where it was highlighted that the challenge lies in terms of implementation due to multiple sub-populations in cooperative neuro-evolution. We note that conventional memetic algorithms feature a single population and hence local search incorporation is relatively easier when compared to several decomposed sub-populations in cooperative neuro-evolution.

We employ a similar strategy for memetic cooperative neuro-evolution presented by Chandra et. al [9] where the best individuals from all the respective sub-populations are concatenated after a given number of cycles for local refinement. Note that a *cycle* refers to the completion of evolution of the respective sub-populations in a round-robin fashion. We employ problem decomposition in cooperative neuro-evolution that is based on neuron level decomposition (CNE-NL) [10]. Each neuron acts as a reference for a subcomponent that contains all the weight connections from the previous layer as shown in Figure 1. The subcomponents are created from reference to all the neurons in the hidden and output layers. These subcomponents are implemented as *sub-populations*.

We present the implementation details of memetic cooperative neuro-evolution for feedforward networks (MCNE) that is given in Algorithm 1 and shown in

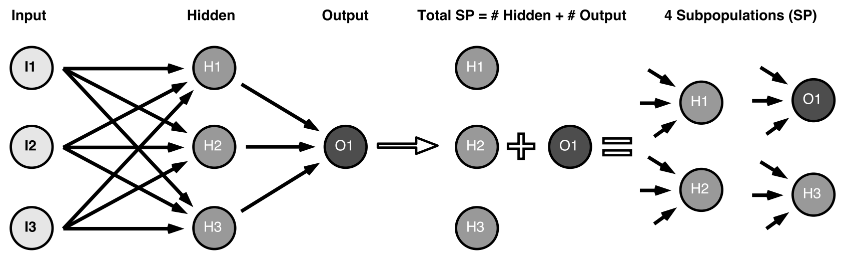


Fig. 1. Neuron level decomposition: a feedforward neural network with 3 input neurons, 3 hidden and 1 output neuron. Neurons in the hidden and output layers act as reference points for new sub-populations, hence the creation of several sub-populations after decomposition.

Figure 1. In Step 1, the algorithm begins by configuring the number of subcomponents needed which is determined by the number of h and o neurons used. The function `rand`(L,s,μ,$\overline{w}$,$\underline{w}$) then initializes the subcomponents in L with a population size of μ containing randomized values between $\underline{w}$ and $\overline{w}$, respectively. Then follows the evaluation phase in `eval`(L,s) where the fitness values of the sub-population individuals are computed. This initialization operation cost also contributes to the total number of function evaluations, Γ_{CC}.

The loop on lines 7 - 16 holds the entire part of evolution that encompasses cooperative neuro-evolution with local refinement by gradient descent which is implemented through back-propagation. The algorithm halts if the specified minimum error ε_t, or if the maximum function evaluations Γ_{max} has been reached. Global search is employed in Step 3 where the sub-populations are evolved in a round-robin fashion using evolutionary operators given by the designated evolutionary algorithm. Hence, the subcomponent D_j is evaluated through function, `evolve`(b,D_j,γ,α) which returns a training error given by the *root-mean-squared-error (RMSE)*. The total number of function evaluations Γ_{CC} for is then updated. This step is repeated according to the local search frequency lsf.

After completion of the evolution *cycle*, the meme δ^* that contains the fittest individuals from each sub-population in L is retrieved and concatenated from a call to function `getbestsolution`(L,s). This calls the local refinement function `bp`(δ^*,λ) shown in Step 4. This is executed for lsi number of iterations where the refined meme δ^* is then broken up in `replaceworst`(L,s,δ^*,b) as shown in line 16 and returned in order to be encoded into the respective sub-populations L, replacing only the individuals with the lowest fitness. Finally, the total number of function evaluations Γ_{CC} is updated in consideration of the lsi. Although gradient descent has been used for local refinement, the proposed algorithm is flexible, and hence other local refinement procedures can also be used.

Algorithm 1: `mcnefnn`$(i, h, o, \alpha, \mu, \Gamma_{max}, \lambda, \varepsilon, lsf, lsi, \delta^*)$

```
Step 1: Initialization
$s$ = h + o;
for $k \in \{1,..,h\}$ & $k \in \{1,..,o\}$ do
    $L_k$ += new $C$;
rand($\mu, \overline{w}, \underline{w}, L, s$); $b$ = eval($L, s$); $\Gamma_{CC} = \mu$ ;
Step 2: Evolution
while $\varepsilon_t > \varepsilon_{min}$ & $\Gamma_{CC} < \Gamma_{max}$ do
    Step 3: Global Search
    for $k \in \{1,..,lsi\}$ (Cycles) do
        for $j \in \{1,..,s\}$ (Subcomponents) do
            $b$ = evolve($b, L_j, \gamma, \alpha$);
            $\Gamma_{CC}$ += $\mu$ x ($\gamma$ + 1);
    $\delta^*$ = getbestsolution($L, s$);
    Step 4: Local Refinement
    for $k \in \{1,..,lsi\}$ do
        $\varepsilon_t$ = bp($\delta^*, \lambda$);
    replaceworst($L, s, \delta^*, b$); $\Gamma_{CC}$ += $lsi$;
```

Table 1. Variables

Variable	Description	Variable	Description
α	species mutation rate	ε_t	test error.
μ	the population size	lsf	local search frequency.
Γ_{max}	max function evaluations.	lsi	local search intensity.
Γ_{CC}	total algorithm evaluations.	δ^*	best meme.
λ	backpropagation learning rate.	L	decomposition components set.
γ	subcomponent optimization time	C	decomposition component.
i	number of input neurons.	s	L size.
h	number of hidden neurons.	$\overline{w}$	upper weight boundary.
o	number of output neurons.	$\underline{w}$	lower weight boundary.
ε_{min}	minimum error needed.		

3 Simulation and Analysis

This section presents an experimental study on the performance of MCNE for chaotic time series problems. We first provide details of the parameters used in the implementation of the algorithms and then present experimental design.

The sub-populations in cooperative neuro-evolution employ a pool of 200 individuals (μ) that feature the G3-PCX evolutionary algorithm (generalized generation gap with parent-centric crossover) [11]. It employs a mating pool size of 2 offspring and 2 parents. The respective sub-populations are initialized within $\{\underline{w}$:-5, $\overline{w}$:5$\}$. We use fixed local search intensity ($lsi = 200$) and local search frequency ($lsf = 10$) that gave good performance in trial experiments. Additionally, backpropagation (gradient descent) employs a learning rate of $\lambda = 0.1$.

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

The embedding dimension D determines the number of input neurons while the number of hidden neurons are varied (from 3 to 9) in order to test for robustness and scalability. The number of function evaluations (Γ_{max}) was set at 5000. The ($RMSE$) and normalized mean squared error ($NMSE$) given in Equations 1 and 2, provide the main performance measures. Note that y_i refers to observed data, $\hat{y}_i$ the predicted data, $\bar{y}_i$ the average of observed data, and N the size of the data.

The datasets employed consist of simulated (Mackey-Glass) and real-world (Sunspot, Laser, and ACI finance) chaotic time series problems. The ACI finance dataset contains closing stock prices of ACI Worldwide Inc. from December 2006 to February 2010 (800 data points [19]). The Laser dataset [24] contains 1000 data points along with the Mackey-Glass [16] while the Sunspot problem consists of 2000 data points [5].

Each problem is divided into a 60-20-20 structure to provide training, validation and test set, respectively. The Taken's embedding theorem [23] is employed to reconstruct original time series data into a usable phase space for training the neural network. The time-lag T determines the interval at which the data points are to be picked while the embedding dimension D, specifies the size of the sliding window. The reconstruction settings for this study are as follows; Mackey-Glass and Laser $\{D{:}3, T{:}2\}$, Sunspot $\{D{:}5, T{:}3\}$, and ACI $\{D{:}5, T{:}2\}$.

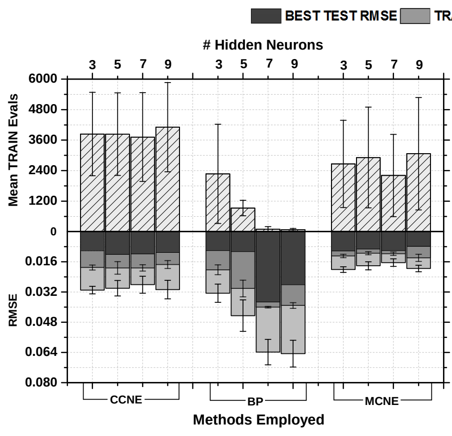
3.1 Results

The results are presented in Figure 2 that highlights the training time (function evaluations) and generalization performance (RMSE) of the given methods. The respective histograms show mean and error bars from with 95 percent confidence interval from 50 independent experimental runs. In Figure 2 (a), MCNE shows significant convergence time improvement over CNE, with the exception of BP. The confidence interval of MCNE is the best for the different number of hidden neurons when we consider the RMSE which shows that it has been the most robust method for this problem. Moreover, MCNE has achieved the highest accuracy in prediction (with lowest values of RMSE).

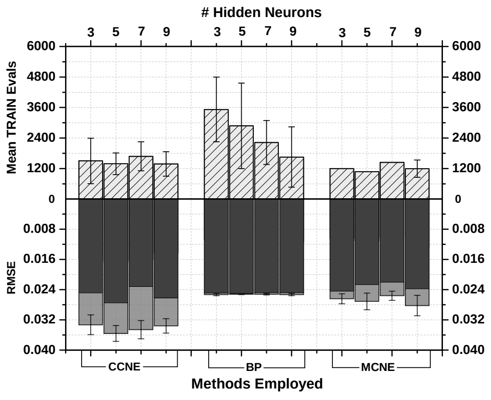
MCNE has shown to get the lowest training time for the ACI-finance problem shown in Figure 2(b). It also achieves best prediction accuracy in terms of lowest RMSE for Sunspot, Mackey-Glass, and Laser problems. MCNE achieves a competing performance with BP for prediction performance for ACI-finance problem

Table 2. Comparison with related methods from the literature

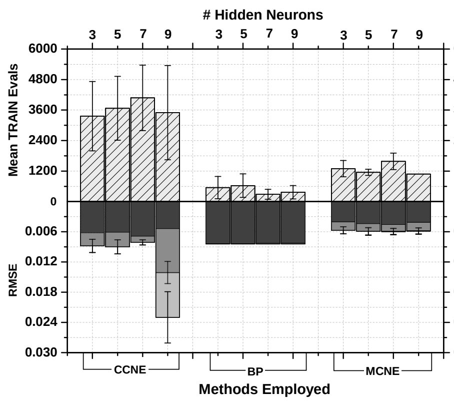
Problem	Prediction Method	RMSE	NMSE
Sunspot [5]	RBF-OLS (2006) [12]		46.0E-03
	LLNF-LoLiMot (2006) [12]		32.0E-03
	SL-CCRNN (2012) [8]	1.66E-02	1.47E-03
	NL-CCRNN (2012) [8]	2.60E-02	3.62E-03
	MO-CCFNN-T=2 (2014) [6]	1.84E-02	1.02E-03
	MO-CCFNN-T=3 (2014) [6]	1.81E-02	0.998E-03
	CICC-RNN (2015) [7]	1.57E-02	1.31E-03
	CCFNN-CSFR (2016) [18]	1.58E-02	**0.756E-03**
	MCNE	**1.01E-02**	1.24E-03
ACI [19]	CICC-RNN (2009) [7]	1.92E-02	
	MO-CCFNN-T=2 (2014) [6]	1.94E-02	
	MO-CCFNN-T=3 (2014) [6]	1.47E-02	
	CCFNN-CSFR (2016) [18]	**1.34E-02**	**0.995E-03**
	MCNE	2.21E-02	1.139E-03
Mackey [16]	RBF-OLS (2006) [12]	**1.02E-03**	460E-04
	LLNF-LoLiMot (2006) [12]	0.961E-03	320E-04
	PS0 (2009) [15]	21.0E-03	
	Neural fuzzy network and DE (2009) [15]	16.2E-03	
	Neural fuzzy network and GA (2009) [15]	16.3E-03	
	SL-CCRNN (2012) [8]	6.33E-03	2.79E-04
	NL-CCRNN (2012) [8]	8.28E-03	4.77E-04
	MO-CCFNN-T=2 (2014) [6]	3.84E-03	0.28E-04
	MO-CCFNN-T=3 (2014) [6]	3.77E-03	0.27E-04
	CICC-RNN (2015) [7]	3.99E-03	1.11E-04
	CCFNN-CSFR (2016) [18]	2.90E-03	**0.016E-04**
	MCNE	4.57E-03	1.23E-04
Laser [24]	RNN-BPTT (2002) [3]		1.54E-02
	RNN-CBPTT (2002) [3]		2.24E-02
	Echo State Network (SN) with IS 0.5 (2011) [22]		9.83E-02
	Echo State Network (SN) with IS 1 (2011) [22]		10.58E-02
	MCNE	**2.32E-02**	**0.589E-02**



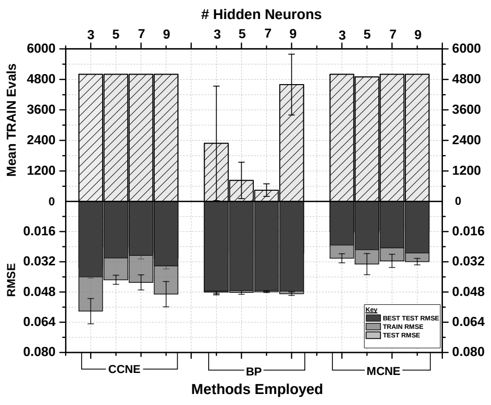
(a) Sunspot [5]



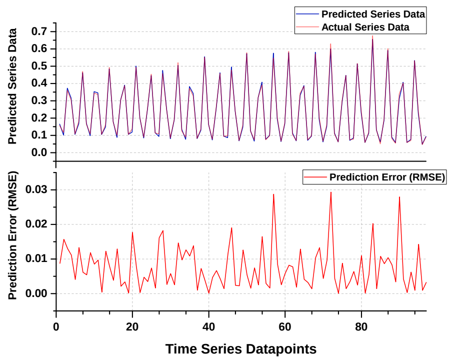
(b) ACI World [19]



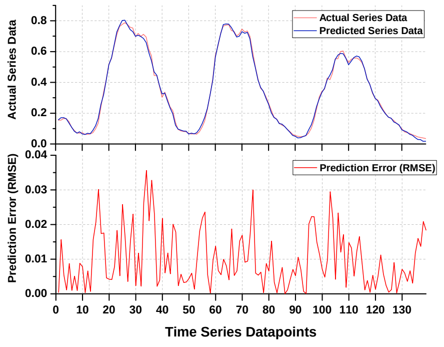
(c) Mackey Glass [16]



(d) Laser [24]



(e) Laser prediction performance



(f) Sunspot prediction performance

Fig. 2. The training time (function evaluations) and generalization performance (RMSE) of the given methods for the Sunspot, ACI finance, Mackey-Glass and Laser time series. Note that the values below the x-axis are not negative. Additionally, (e) and (f) shows the prediction accuracy of MCNE for a single test run for the Sunspot and Laser problems.

where BP takes most training time when compared to the other methods. This case shows the inefficiency of BP convergence although it is perceived as a faster method when compared to evolutionary techniques. In terms of robustness, 5 hidden neurons provide the best convergence and prediction performance for all the problems.

Table 2 presents a comparison with related methods from the literature. We note that MCNE shows the best performance for Sunspot and Laser problems however the algorithm did not perform very well for the ACI finance and Mackey problems. One of the problems with direct comparison with other methods is the difference in training time and related parameters of the experimental design. We employed maximum of 5000 function evaluations for training while compared to Nand and Chandra [18] (CSFR in Table 2) used 50,000 function evaluations. Hence, more training time could be attributed to better performance. Additionally, recurrent neural networks seem to be better suited for time series prediction [8].

3.2 Discussion

The results show that the proposed method performs very well on the real-world problems (Sunspot and Laser) when compared to standalone methods. The improvement can be credited to solution transfer between global and local search, especially through the use of gradient information. We note that extra emphasis on global search increases computational costs that produce lower accuracy since neuro-evolution is essentially a black-box training method. Real world problems are often non-linear and noisy thus the diversity of solutions through exchange between local and global search has been beneficial. A major advantage of local refinement has been the decrease in the number of function evaluations in reaching a promising solution. The results show that local refinement in evolution speeds convergence and produces better accuracy for chaotic time series prediction.

Although the method has shown to perform better than its counterparts, the results face some challenges when compared to some of the methods from the literature. We note that it is not fair for a straight-forward comparison with results from the literature as some of the methods rely on different approaches that include hybrid methods, ensembles, and feature extraction. However, an extensive study on the balance between the local search frequency and intensity in our proposed method is needed for further improvement.

4 Conclusion and Future Work

We presented a memetic cooperative neuro-evolution method where backpropagation was used for local refinement. The method was applied to selected problems in chaotic time series prediction and the results showed promising when compared to related methods in the literature. This was mainly for real-world application problems that feature noise and are chaotic in nature. Cooperative neuro-evolution provided diversification while backpropagation refined the promising solutions that lead to rapid convergence. We highlight that the proposed method has been feasible even with limited training time.

The proposed method is flexible and can be easily adapted to incorporate other local search methods. A group of local search methods can also be used that can exploit different regions of search space at different stages of the learning process.

A limitation of this study has been the lack of extensive study on the balance between diversification and intensification. Hence, future work can focus on this in the context of time series prediction. Furthermore, the results motivate real-world applications to domains that range from finance to climate change. Further improvements in the results could be achieved with different local refinement procedures. Moreover, the method can be extended to multi-step-ahead time series prediction.

References

[1] M. Alsmadi, K. Omar, S. Noah, and I. Almarashdeh. A hybrid memetic algorithm with back-propagation classifier for fish classification based on robust features extraction from plgf and shape measurements. *Information Technology Journal*, 10(5):944–954, 2011. ISSN 1812-5638.

[2] A. Arab and A. Alfi. An adaptive gradient descent-based local search in memetic algorithm applied to optimal controller design. *Information Sciences*, 299:117–142, 2015. ISSN 0020-0255.

[3] R. Boné, M. Crucianu, and J.-P. A. de Beauville. Learning long-term dependencies by the selective addition of time-delayed connections to recurrent neural networks. *Neurocomputing*, 48(1):251–266, 2002.

[4] P. Castillo, M. Arenas, J. Castellano, J. Merelo, A. Prieto, V. Rivas, and G. Romero. Lamarckian evolution and the baldwin effect in evolutionary neural networks. *arXiv preprint cs/0603004*, 2006.

[5] S. W. D. Center. The international sunspot number. *International Sunspot Number Monthly Bulletin and online catalogue*, 1834 - 2001.

[6] S. Chand and R. Chandra. Multi-objective cooperative coevolution of neural networks for time series prediction. In *Neural Networks (IJCNN), 2014 International Joint Conference on*, pages 190–197. IEEE, 2014.

[7] R. Chandra. Competition and collaboration in cooperative coevolution of elman recurrent neural networks for time-series prediction. *Neural Networks and Learning Systems, IEEE Transactions on*, 26(12):3123–3136, 2015.

[8] R. Chandra and M. Zhang. Cooperative coevolution of elman recurrent neural networks for chaotic time series prediction. *Neurocomputing*, 86:116–123, 2012. ISSN 0925-2312.

[9] R. Chandra, M. Frean, and M. Zhang. Crossover-based local search in cooperative co-evolutionary feedforward neural networks. *Applied Soft Computing*, 12(9):2924–2932, 2012. ISSN 1568-4946.

[10] R. Chandra, M. Frean, and M. Zhang. On the issue of separability for problem decomposition in cooperative neuro-evolution. *Neurocomputing*, 87:33–40, 2012.

[11] K. Deb, A. Anand, and D. Joshi. A computationally efficient evolutionary algorithm for real-parameter optimization. *Evol. Comput.*, 10(4):371–395, 2002. ISSN 1063-6560.

[12] A. Gholipour, B. N. Araabi, and C. Lucas. Predicting chaotic time series using neural and neurofuzzy models: a comparative study. *neural processing letters*, 24 (3):217–239, 2006.

[13] S. A. Kazarlis, S. E. Papadakis, J. Theocharis, and V. Petridis. Microgenetic algorithms as generalized hill-climbing operators for ga optimization. *Evolutionary Computation, IEEE Transactions on*, 5(3):204–217, 2001. ISSN 1089-778X.

[14] B. Li, Y.-S. Ong, M. N. Le, and C. K. Goh. Memetic gradient search. In *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 2894–2901. IEEE, 2008. ISBN 1424418224.

[15] 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. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 39(1):55–68, 2009.

[16] M. C. Mackey and L. Glass. Oscillation and chaos in physiological control systems. *Science*, 197(4300):287–289, 1977.

[17] P. Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Technical Report 826, Caltech Concurrent Computation Program, 1989.

[18] R. Nand and R. Chandra. *Artificial Life and Computational Intelligence: Second Australasian Conference, ACALCI 2016, Canberra, ACT, Australia, February 2-5, 2016, Proceedings*, chapter Coevolutionary Feature Selection and Reconstruction in Neuro-Evolution for Time Series Prediction, pages 285–297. Springer International Publishing, Cham, 2016. ISBN 978-3-319-28270-1. doi: 10.1007/978-3-319-28270-1_24.

[19] NASDAQ. URL http://www.nasdaq.com/symbol/aciw/stock-chart.

[20] Y. S. Ong and A. J. Keane. Meta-lamarckian learning in memetic algorithms. *Evolutionary Computation, IEEE Transactions on*, 8(2):99–110, 2004. ISSN 1089-778X.

[21] M. A. Potter and K. A. De Jong. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary computation*, 8(1):1–29, 2000.

[22] A. Rodan and P. Tiňo. Minimum complexity echo state network. *Neural Networks, IEEE Transactions on*, 22(1):131–144, 2011.

[23] F. Takens. Detecting strange attractors in turbulence. In *Dynamical Systems and Turbulence, Warwick 1980*, Lecture Notes in Mathematics, pages 366–381. 1981.

[24] A. S. Weigend and N. A. Gershenfeld. Laser problem dataset: The santa fe time series competition data, 1994. URL http://www-psych.stanford.edu/~andreas/Time-Series/SantaFe.html.

[25] X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.