Breaking the Stereotypical Dogma of Artificial Neural Networks with Cartesian Genetic Programming

Gul Muhammad Khan and Arbab Masood Ahmad

Abstract This chapter presents the work done in the field of Cartesian Genetic Programming evolved Artificial Neural Networks (CGPANN). Three types of CGPANN are presented, the Feed-forward CGPANN (FFCGPAN), Recurrent CGPANN and the CGPANN that has developmental plasticity, also called Plastic CGPANN or PCGPANN. Each of these networks is explained with the help of diagrams. Performance results obtained for a number of benchmark problems using these networks are illustrated with the help of tables. Artificial Neural Networks (ANNs) suffer from the dilemma of how to select complexity of the network for a specific task, what should be the pattern of inter-connectivity, and in case of feedback, what topology will produce the best possible results. Cartesian Genetic Programming (CGP) offers the ability to select not only the desired network complexity but also the inter-connectivity patterns, topology of feedback systems, and above all, decides which input parameters should be weighted more or less and which one to be neglected. In this chapter we discuss how CGP is used to evolve the architecture of Neural Networks for optimum network and characteristics. Don't you want a system that designs everything for you? That helps you select the optimal network, the inter-connectivity, the topology, the complexity, input parameters selection and input sensitivity? If yes, then CGP evolved Artificial Neural Network (CGPANN) and CGP evolved Recurrent Neural Network (CGPRNN) is the answer to your questions.

G.M. Khan (✉)
Department of Electrical Engineering, University of Engineering and Technology, Peshawar, Pakistan
e-mail: gk502@uetpeshawar.edu.pk

A.M. Ahmad
Department of Computer Systems Engineering, University of Engineering and Technology, Peshawar, Pakistan
e-mail: arbabmasood@uetpeshawar.edu.pk

S. Stepney and A. Adamatzky (eds.), *Inspired by Nature*, Emergence, Complexity and Computation 28, https://doi.org/10.1007/978-3-319-67997-6_10

1 Artificial Neural Networks

Artificial Neural Network (ANN) is a powerful tool for non-linear mapping between input and output. It is the ultimate solution when all the traditional algorithms fail. It provides a parametric expression for the system whose mathematical model for functionality is unknown. Despite successful application of neural networks to a variety of problems, they still have some limitations. One of the most common limitations is associated with neural network training. The back-propagation learning algorithm cannot guarantee an optimal solution. Back propagation has the tendency to get stuck-up at a sub-optimal point, and might never come out of it. Much research has been done to tackle this problem. Evolutionary methods can provide solution to this problem. They do so by producing multiple solutions at any one point, thus if one of them is stuck-up at suboptimal point the other might be in closer vicinity to the global optimum and will thus cause the network to reach the global optimum. In real-world applications, the back-propagation algorithm might converge to a set of sub-optimal weights from which it cannot escape. As a result, the neural network is often unable to find a desirable solution to a problem at hand. To get the benefits of both ANN and evolutionary methods researchers tried a hybrid of both these methods [33]. Another difficulty is related to selecting an optimal topology for the neural network. The right network architecture for a particular problem is often chosen by means of heuristics, and designing a neural network topology is still more of art than engineering. Genetic algorithm [25] and genetic programming [28] are effective optimization techniques that can guide both weight and topology selection.

2 Neuro-Evolution

The process of evolving various parameters of neural network is termed Neuro-evolution. Unlike backpropagation algorithm which is used to train only weights of the network to obtain the desired optimum characteristics, evolutionary techniques can train the network topology and even the learning rules. In case of evolving connection weights, we perform the following steps:

1. Encode the connection weights of each individual neural network into chromosomes.
2. Calculate the error function and determine the individual‘s fitness.
3. Reproduce children based on selection criteria.
4. Apply genetic operators.

Success or failure of an application is largely determined by the network architecture (i.e. the number of neurons and their interconnections). As the network architecture is usually decided by trial and error, a good algorithm is required to automatically design an efficient architecture for each particular application. Genetic algorithms may well be suited for this task. The basic idea behind evolving a suitable network

architecture is to conduct a genetic search in a population of possible architectures. We must first choose a method of encoding a network's architecture into a chromosome. There are two types of encoding schemes:

Direct Encoding: When all the information of the network is represented directly by genes in the code, it is referred to as Direct Encoding Scheme. In this case the network has a one to one relationship between genotype and phenotype.

Indirect Encoding: In this type of encoding the genes don't represent the network directly, and show only the indirect function responsible for generation of network parameters [5]. This is biologically more plausible, as according to the findings of neuroscience it is impossible for genetic information to be encoded in humans to specify the whole nervous system directly. It is computationally more expensive, since it doesn't have any clue of the targeted application for which the network is developed.

3 CGP Evolved Artificial Neural Network (CGPANN)

We have used CGP to introduce four different ways of evolving neural networks that are as follows:

- Feed-forward CGP evolved ANN (FCGPANN)
- Recurrent CGP evolved ANN (RCGPANN)
- Plastic CGP evolved ANN (PCGPANN)
- Plastic Recurrent CGPANN (PRCGPANN).

3.1 Feed-Forward CGP Evolved ANN (FCGPANN)

In the first case, CGP is transformed to a feed-forward neural network by considering each node as a neuron, and providing each connection with a weight. The neurons of such a network are arranged in Cartesian format with rows and columns inspired by original CGP architecture, and later on restricted to a single row mostly giving the network an ability to create infinite graphs/topologies. Each neuron in the network can take connection from either a previous neuron or from the system input. Not all neurons are necessarily connected with each other or with system inputs, this provides the network with an ability to continuously evolve its complexity along with the weights. All the network parameters are represented by a string of numbers called genotype. The number of active neurons (connected from inputs to outputs) varies from generation to generation subject to the genotype selection. Output of any neuron or a system input can be a candidate for the systems output selection. The ultimate system functionality is identified by interconnecting neurons from output to input. Since CGP works best with mutation, thus only mutation operator is

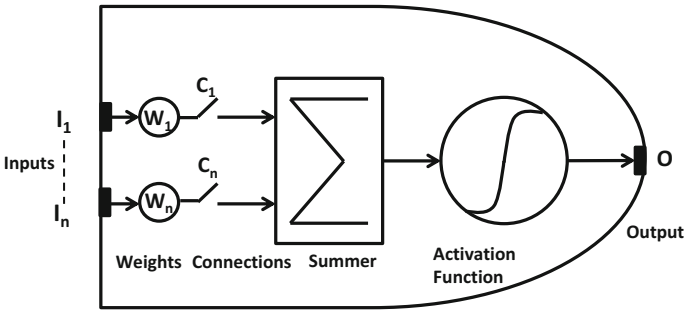

Fig. 1 A CGPANN node containing the inputs, weights, switches, summer and activation function

explored in all the variants of neural networks introduced in this work. Mutation operator specifies the percentage of genes to be mutated during the process of evolution. A gene can be an input connection to a node, a weight, a switch or a neuron function. These genes for a single node are shown in Fig. 1. An FCGPANN genotype for arity 2 (two inputs per node) is represented by the following expression: $FI_1W_1C_1I_2W_2C_2, FI_1W_1C_1I_2W_2C_2 \ldots, O_1, O_2 \ldots O_n$. Where **F** is the activation function chosen randomly from a list of different type of nonlinear functions. The two most popular functions, also used in this work are the Log-sigmoid and the tangent hyperbolic function, **I** represents output of a node or a system input, connected to the node under consideration, **W** is the weight being multiplied with a node input and **C** is an optional ON/OFF switch. All these genes can get only certain allowed values depending on the type of network. The network representation of a genotype is called a phenotype. Figure 2 shows a typical FCGPANN phenotype with its associated genotype. We have the option to evolve, all the parameters, or can fix one or two and evolve others.

FCGPANN was initially tested for its speed of learning, and evaluated against the previously introduced neuro-evolutionary techniques on benchmarks such as single and double Pole balancing [19]. Table 1 shows the superior performance of FCGPANN and RCGPANN (see next section) in comparison to the other neuroevolutionary techniques evaluated for speed of learning on single pole balancing problem. Table 2 shows the performance of FCGPANN and RCGPANN compared to other techniques, for Markovian and non-Markovian cases of double pole balancing task, where a Markovian process can be defined as the one in which the conditional probability distribution of the future state depends only on the present state and not on the past history. The figures show average number of evaluations needed to achieve the target objective of balancing the poles for a specific bench marked time interval. FCGPANN is explored in a range of applications including: breast cancer detection, prediction of foreign currency exchange rates, Load forecasting, Internet multimedia traffic management, cloud resource estimation, solar irradiance prediction, wind power forecasting and arrhythmia detection [2, 13, 14, 16, 19, 24, 30]. FCGPANN outperformed all the previously introduced techniques as highlighted in

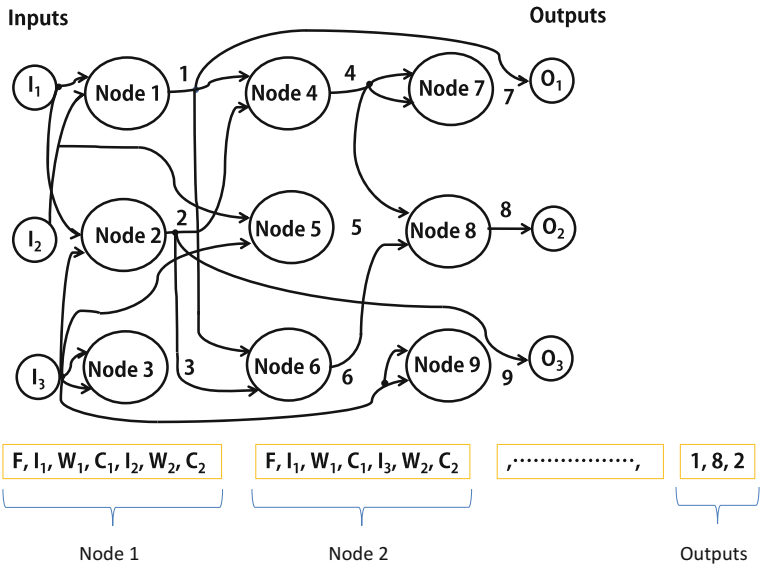

Fig. 2 A CGPANN phenotype with its corresponding genotype

Table 1 Comparison of CGPANN with other neuro-evolutionary algorithms in terms of average number of evaluations required to solve the single pole balancing task

Method	Markovian	Non-Markovian
Conventional Neuro-Evolution (CNE) [36]	352	724
Symbiotic, Adaptive Neural Evolution (SANE) [23]	302	1212
Enforced sub-population (ESP) [7]	289	589
Neuro-Evolution of Augmenting Topologies (NEAT) [35]	743	1523
Cooperative synapse neuroevolution (CoSyNE) [6]	98	127
FCGPANN [19]	21	–
RCGPANN [19]	17	55

the literature. Table 3 shows the comparative results for the mean accuracy in breast cancer detection with fine needle aspiration (FNA), using different algorithms. It can be seen that FCGPANN outperformed the other methods [19]. Another important area in which FCGPANN was successfully applied is the prediction of foreign currency exchange rates [24]. Table 4 shows the comparative results for prediction of foreign currency exchange rates using FCGPANN and other methods.

Marketing a product requires good knowledge about the demands of customers, especially in the case of food products. FCGPANN provides efficient method to predict market trends [1].

Table 2 Comparison of CGPANN with other neuro-evolutionary algorithms in terms of average number of evaluations required to solve the double pole balancing task

Method	Markovian	Non-Markovian	
	Standard fitness	Standard fitness	Damping fitness
Conventional Neuro-Evolution (CNE) [36]	22 100	76 906	87 623
Symbiotic, Adaptive Neural Evolution (SANE) [23]	12 600	262 700	451 612
Enforced sub-population (ESP) [7]	3 800	7 374	26 342
Neuro-Evolution of Augmenting Topologies (NEAT) [35]	3 600	–	6 929
Cooperative synapse neuroevolution (CoSyNE) [6]	954	1 294	3416
FCGPANN [19]	77	–	–
RCGPANN [19]	129	163	387

Table 3 Comparison of Mean Absolute Percentage Errors (MAPE) obtained using various classification methods using the processed FNA data from WDBC that contains 30 features

No.	Method	Mean (MAPE)
1	Multi-Layer Perceptron (MLP) [8]	95.56
2	Fisher Linear Discriminant Analysis (FLDA)/MLP [8]	90.92
3	Principle Component Analysis (PCA)/MLP [8]	92.02
4	Genetic Programming (GP/MDC) [8]	96.58
5	Evolutionary Neural Network (ENN) [10]	95.6
6	FCGPANN [19]	97 for Type-I and 98.5 for Type-II

Table 4 Comparison between the MAPE of other well known methods and that of FCGPANN, for predicting foreign exchange rates

Network	MAPE (%)
Hidden Markov model (HMM) [29]	1.928
ARIMA [3]	1.6108
Regression model [29]	1.9
CART model	1.62
Neural network model [9]	1.61
FCGPANN [24]	1.148

3.2 Recurrent CGPANN (RCGPANN)

The second type of CGPANN is the Recurrent CGPANN (RCGPANN). These networks are more suitable for modeling systems that are dynamic and nonlinear. This network is a modification to one of the earliest networks, the Jordan's network [11].

In the Jordan's network there are state inputs that are equal in number to the outputs. These inputs are fed by the outputs through unit weights. The state inputs are present only at the input layer. Learning of these networks take place by changing the weights of connections between input layer and the hidden layer and, the hidden and the output layer. In RCGPANN unlike the Jordan's network the state inputs can be connected, not necessarily to the first layer but to any layer. These additional inputs also have the activation functions. Figures 3 and 4 show a typical RCGPANN neuron, and the RCGPANN genotype and phenotype respectively. Here I_1 and I_2 are the normal inputs while R is a state input. Initial value of the R input to the system is considered zero. Output is taken from Node 6 as evident from genotype and the corresponding phenotype, and node 6 takes input from node 3 and input I_2 only. Node 4 and 5 do not contribute to the output and are termed inactive nodes, while 3 and 6 are active nodes as they contribute to the output. Following are the outputs of active nodes:

$$\psi_3 = tanh(I_1 \cdot W_{13} + I_2 \cdot W_{23} + R \cdot W_{R3})$$

$$\psi_6 = tanh(\psi_3 \cdot W_{36} + I_2 \cdot W_{26} + R \cdot W_{R6})$$

where I_1 and I_2 are the normal inputs to the system, R is the state input, W_{mn} is the weight of connection between system-input/node m and n, ψ_m is the output of node m. Figure 5 shows the case when all the outputs are presented as feedback

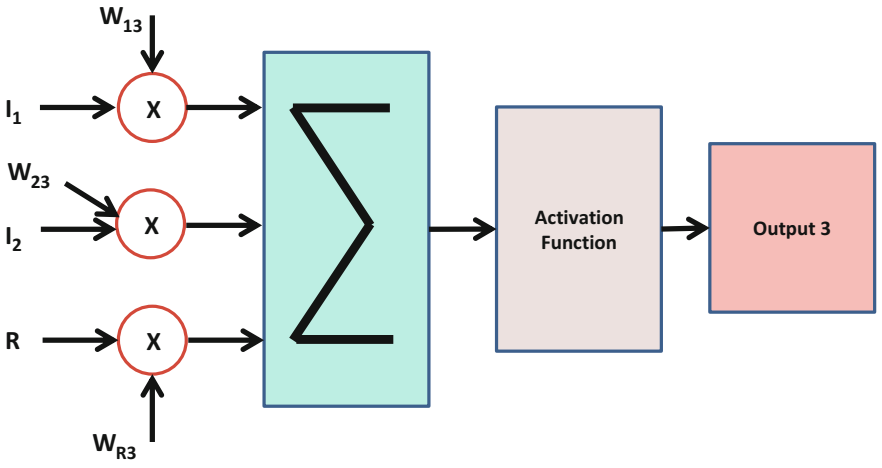

Fig. 3 A typical RCGPANN neuron

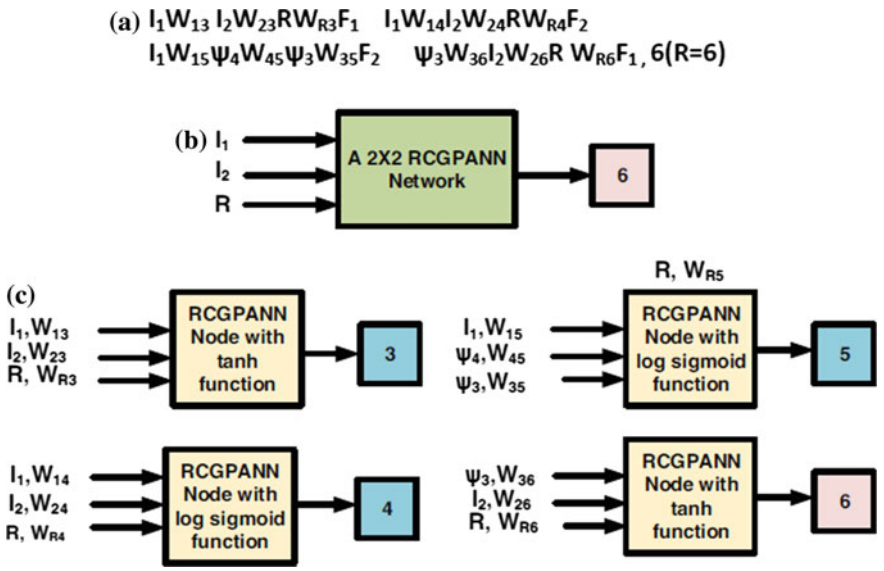

Fig. 4 Figure showing **a** RCGPANN genotype, **b** 2 × 2 RCGPANN Network and **c** corresponding to the phenotype

inputs to the neurons for selection. The output is averaged to find the desired output of the system. RCGPANN was also tested initially for its speed of learning similar to FCGPANN on both single and double pole balancing for both markovian and nonmorkovian cases. Its performance relative to other neuroevolutionary techniques for Markovian and non-Markovian cases explored on single pole balancing task is shown in Table 1 and that on double pole is shown in Table 2. The numbers in the tables represent the required average number of evaluations to find the desired pole balancing behaviour. The results in these two tables clearly show the superiority of FCGPANN and RCGPANN in Markovian case and that of RCGPANN in the non-Markovian case.

FCGPANN and RCGPANN are also tested for their generalization ability. Table 5 shows generalization of FCGPANN and RCGPANN: average number of random cart initializations (out of 625) that can be balanced for a desired number (benchmark) of time-steps. Table 6 presents the generalization ability of various neuroevolutionary algorithms for the double pole balancing scenario for non-Markovian case using the damping fitness function. It is observed that the RCGPANN scored 335.84 out of 625 for 50 independently evolved genotypes exhibiting greater generalization ability as compared to other techniques presented to date. Recurrent CGPANN has been successfully applied to a number of applications including: Load forecasting, foreign currency exchange rates, bandwidth management and estimation [16, 17, 31]. RCGPANN has been successfully applied to electrical load prediction for a complete year and also for different seasons of the year, enabling efficient utilization of

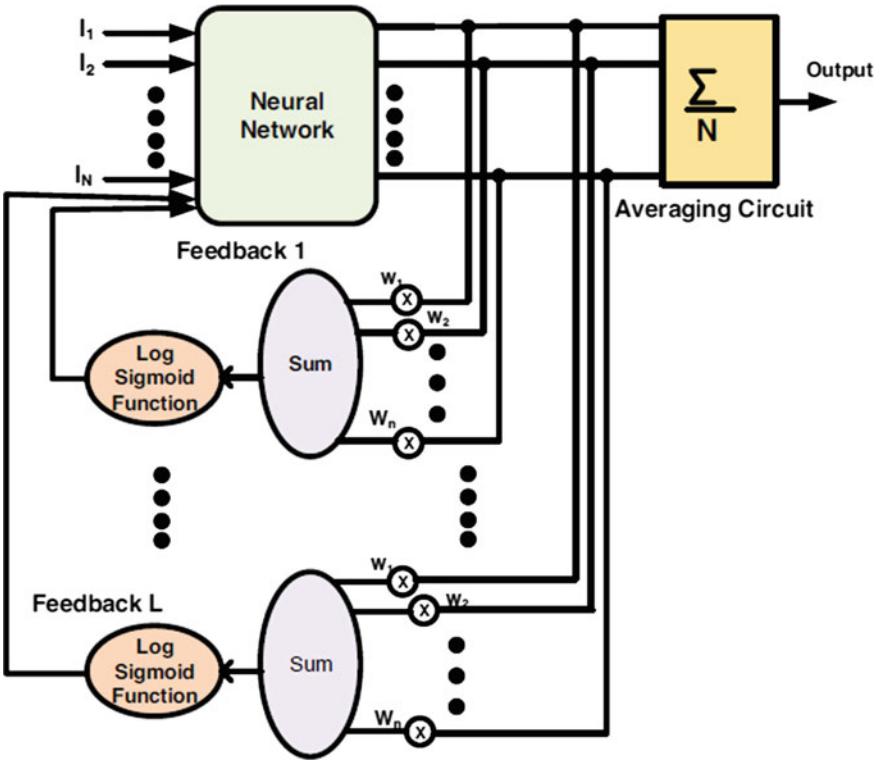

Fig. 5 RCGPANN phenotype with full feedback having all outputs available for feedback

electricity management [14]. Table 7 compares the performance of RCGPANN with other contemporary methods in terms of Mean Absolute Percentage Error (MAPE), for the load forecasting task. Bandwidth allocation for communication channels has always been a challenge for the engineers. Recurrent CGPANN has been successfully applied to predict the size of next MPEG4 video frame based on the estimate of the last ten frames [16]. Table 8 shows the comparative results of next frame prediction in terms of overall error for RCGPANN and other models. The electric load prediction discussed earlier was improved to predict very short term (about half an hour) load [18]. Table 9 shows the performance in terms of MAPE values, for the proposed RCGPANN in comparison to other methods, for predicting very short term electric load. Table 10 shows the comparative results for predicting the foreign currency exchange rate using RCGPANN and other models [31].

Table 5 Generalization of CGPANN: average number of random cart initializations (out of 625) that can be solved

Algorithm type	Markovian		Non-Markovian	
	Single pole	Double pole	Single pole	Double pole
FCGPANN	590	277.38	–	–
RCGPANN	363.94	471.92	294	335.84

Table 6 Comparison of generalization of CGPANN with other neuroevolutionary algorithms for the double pole balancing scenario for Non-Markovian case

Method	Value
CE	300
ESP	289
NEAT	286
RCGPANN	335.84

Table 7 Comparison of RCGPANN with other methods for the load forecasting task

Method	MAPE (%)
Local linear model tree	1.98
Support vector machine	1.93
Autonomous ANN	1.75
Floating search + SVM	1.70
CGPANN	1.71
ANN-back propagation	2.41
GA based adaptive ANN	1.94
RCGPANN	1.56

Table 8 Best overall error comparison in MPEG4 frame size prediction

S.no.	Scheme	Error (%)
1	Recurrent ANN	RMSE = 3.0
2	F-CGPANN	RMSE = 16
3	Laetitia et al. model	RPE = 7.30
4	SARIMA	MARE = 1.37
5	Kalman filter	MARE = 1.4
6	Proposed (RCGPANN)	RMSE = 2.7 MAPE = 1.2

Table 9 Comparison of RCGPANN with other methods in terms of MAPE for very short time electric load forecasting

S.no	Model	MAPE (%)
1	Self-supervised adaptive ANN	0.91
2	FNN for RTLF	0.88
3	ANNSTLF	2
4	RBF forecaster	1.3393
5	Model in [34]	0.66
6	Multiplicative decomposition model	0.7601
7	Seasonal ARIMA model	1.6108
8	Model 1 [22]	1.792
9	Model 2 [22]	1.813
10	RCGPANN (proposed model)	0.43

Table 10 Comparison between the accuracy distribution rates and MAPE of RCGPANN and other models for the foreign currency exchange rate prediction

Network	Accuracy (%)
Multi layer perceptron [21]	72
HFERFM [27]	69.9
AFERFM [27]	81.2
Backpropagation with Bayesian regularization [12]	93.93
RCGPANN (implemented)	98.872

3.3 Plastic CGPANN (PCGPANN)

Plasticity in neural networks has been the characteristic of choice when it comes to applications in dynamic systems due to its comparatively better performance [4, 26, 32]. The improved performance in Plastic neural networks can be attributed to the adaptability of its morphology to environmental stimuli. This is similar to the natural neural system. In this developmental form of CGPANN an additional output gene provides extra features to the system. PCGPANN has the same basic structure as that of CGPANN presented previously. With the addition of an extra output gene, that causes developmental decisions during evaluation process, the CGPANN achieves its plasticity. The decision is made on the basis of output values. The mutation of the genotype is invoked according to a decision function that decides either to invoke mutation or not. The decision function in this case is a threshold function which invokes mutation when the value of output of the CGPANN is higher than the threshold value. The threshold value is selected based on the performance of the model. In this way an unlimited number of phenotypes might be generated from a

single genotype, depending on the system requirement. The network is tested for its performance in diverse learning domains. The genotypes are modified based on the defined set of constraints. The plasticity invokes mutation of genotype at runtime, modifying its genes, producing complex network structures.

3.3.1 PCGPANN Methodology

The PCGPANN generalized approach is demonstrated in Fig. 6. The figure depicts network topology, its parameters i.e. inputs, connections, outputs, the additional output gene and the algorithm of PCGPANN. The system consists of the original CGPANN as explained before. Output from this network is fed into the running sum block. Based on the evolutionary requirements, the number of outputs fed into the summation block can vary. Either half or full number of CGPANN outputs are fed into the summation block. The network is thus named Full Feedback (FFB) or Half Feedback (HFB) network, based on its architecture. Output of the summation block is fed to a decision function that generates either a 0 or a 1 so as to invoke the mutation or not. Depending on the value of the summation block the decision function either invokes mutation or leaves the genotype unchanged. The network is unique in its ability to invoke mutation during run time. The mutation may take place randomly in any of the following network genes: node inputs, weights, outputs or activation functions. The genes are mutated under the given set of constraints.

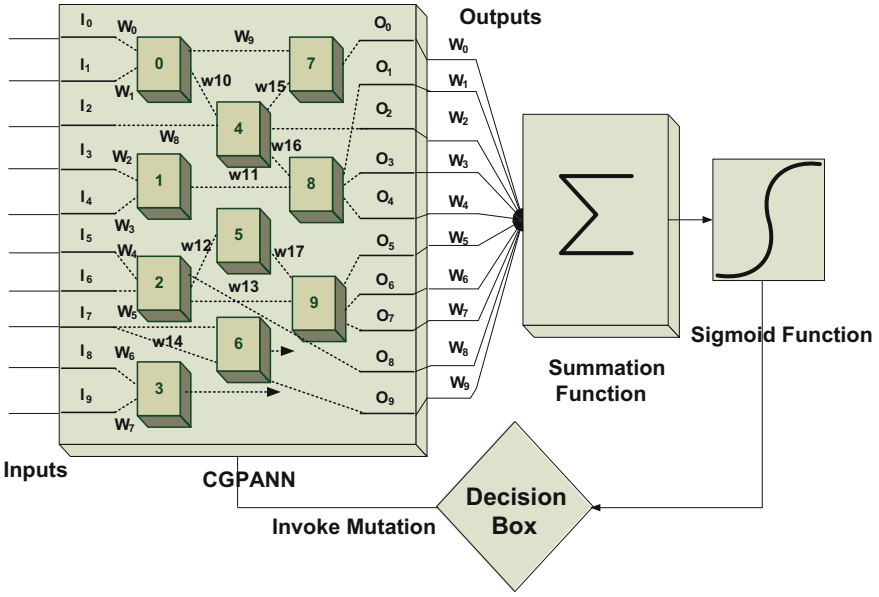

Fig. 6 A Generalized approach of PCGPANN depicting the network topology, attributes: inputs, connections, outputs, the additional output gene and the algorithm

Constraints:

1. If the gene that has to be mutated is the function of a node, then a new function is randomly selected from a list of functions and assigned to the node.
2. If a weight gene has to be mutated, then it is assigned a random value between −1 and +1.
3. If the gene to be mutated happens to be a node input, then depending on the levels-back parameter, output of any randomly selected node on the left of the node under consideration or a network input is connected to it.
4. If an output gene has to be mutated, then it is connected to the output of a randomly selected node or a system input.

In each iteration genes are mutated, the number of mutations depending on runtime mutation rate (Developmental index). The possibility of change in genotype at runtime is dependent on the output from activation function. The decision function is maintained in accordance to the expected range of output from the activation function.

3.3.2 Development in PCGPANN

The plasticity in PCGPANN is based on the decision function. Once developmental index is invoked by the function, the network initiate a step by step process of development under the given set of aforementioned constraints. An example of various possible steps of development in PCGPANN are illustrated in Fig. 7. A change may be invoked in either the input of a node, a function or a weight gene as shown in the figure. The process is highlighted both in the genotype and the phenotype. Figure 7a shows the initial genotype and phenotype. There are two node functions, a sigmoid and a tangent hyperbolic, in the initial genotype. These are the available functions that can be randomly assigned to a node when mutation takes place. The ψ_0s and ψ_1s are network inputs and ψ_2 is the output of the first node and ψ_3 is the system output, while the weights have values in the range $[-1, 1]$. The mathematical representation of the initial network is given in Eq. 1.

$$\psi 3 = Sig\left[0.6\psi 1 + 0.2Tanh(0.2\psi 0 + 0.7\psi 1)\right] \tag{1}$$

In the first case, the function of first node i.e. Tanh is transformed to sigmoid function. The updated genotype and phenotype are presented in Fig. 7b. The mathematical expression for the updated genotype is given in Eq. 2.

$$\psi 3 = Sig\left[0.6\psi 1 + 0.2Sig(0.2\psi 0 + 0.7\psi 1)\right] \tag{2}$$

In the second case, the weight changes from 0.1 to 0.7 i.e. at the first input of the second node as shown in Fig. 7c. The mathematical expression for the updated genotype is given by Eq. 3.

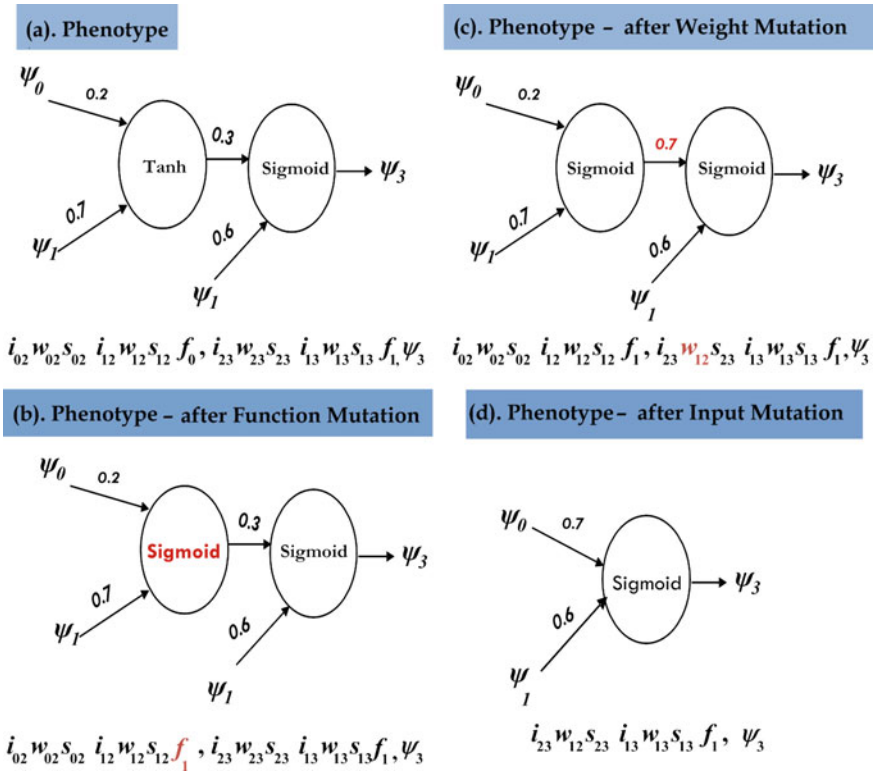

Fig. 7 Demonstration of real time development in PCGPANN

$$\psi 3 = Sig\left[0.6\psi 1 + 0.7Sig(0.2\psi 0 + 0.7\psi 1)\right] \tag{3}$$

In Fig. 7d, the structure is modified when an input to the neuron is altered i.e. the first input to the second node is disconnected from node 1 and connected to a system input. The network attains final expression as given by Eq. 4.

$$\psi 3 = Sig\left[0.6\psi 1 + 0.7\psi 0\right] \tag{4}$$

Similar to FCGPANN and RCGPANN, PCGPANN is also first evaluated for its learning ability and then the generalization of performing in an unknown environment. Table 11 shows the comparative results for PCGPANN and other algorithms, used for single and double pole balancing tasks. The performance is shown in terms

Table 11 Comparison of PCGPANN with other neuroevolutionary algorithms applied on single and double pole balancing task: average number of network evaluations

Method	Single pole	Double pole
Conventional Neuro-Evolution (CNE) [36]	352	22 100
Symbiotic, Adaptive Neural Evolution (SANE) [23]	302	12 600
Enforced sub-population (ESP) [7]	289	3 800
Neuro-Evolution of Augmenting Topologies (NEAT) [35]	743	3 600
Cooperative synapse neuroevolution (CoSyNE) [6]	98	954
FCGPANN	21	77
PCGPANN	104	1 169

Table 12 Average number of random cart-pole initializations (out of 625) that can be solved

Type	Single pole	Double pole
FCGPANN	590	277.38
PCGPANN	456	349

Table 13 Comparison of PCGPANN with other ANNS for the prediction of foreign currency exchange rates

Network	Accuracy
AFERFM	81.2
HFERFM	69.9
Multi layer perceptron	72
Volterra network	76
Back propagation network	62.27
Multi neural network	66.82
CGPANN	98.85
PCGPANN [15]	98.8516

of average number of network evaluations [20]. Table 12 shows generalization of the PCGPANN genotypes in comparison to FCGPANN for both the single and double pole balancing scenarios. Plastic CGPANN has also been successfully applied to evolve a dynamic and robust computational model for efficiently predicting daily foreign currency exchange rates in advance based on past data [15]. Table 13 shows the comparative results of foreign currency exchange rate prediction using PCGPANN and other contemporary methods.

3.4 *Plastic Recurrent Cartesian Genetic Programming Evolved Artificial Neural Network (PRCGPANN)*

Plastic Recurrent Cartesian Genetic Programming Evolved Artificial Neural Network is an online learning approach that incorporates developmental plasticity in Recurrent Neural Networks. Recurrent Neural Networks can process arbitrary sequences of inputs due to their ability to access internal memory. In a Plastic RCGPANN the output gene not only forms the system output but also plays a role in the developmental decision. Output of the system is applied to a decision function to invoke development of the network. Development in the phenotype takes place with the mutation of the genotype in runtime. The recurrent CGPANN has a feedback mechanism in the network, that feeds one or more outputs back to the system. The general approach of PCGPRNN is depicted in Fig. 8. The figure shows the inputs, outputs, connections, recurrent inputs and the output gene that invokes development in the network. The initial network is the original representation of the genotype that changes in response to the output of the system with the passage of time. The decision regarding the development is the reflection of output of the system fed into the sum block and the recurrent paths taken from the CGPANN block associated with the weights, summation and sigmoid function as shown in the figure. If the value obtained from the function is less than defined decision value, development is invoked in the network or otherwise the outputs are monitored without any modification. The uniqueness aspect of the approach is that network changes take place in real time according to the data flow in the network and this provides the plasticity feature to the network. PRCGPANNs invoke changes in the network by changing (or mutating) a node function, an input, a weight or by switching an input in the real time. The learning rules for development in the PCGPRNN are achieved during the process of evolution. A special case is presented here to describe the developmental mechanism in PRCGPANN at run time as shown in Figs. 9, 10 and 11. The system has three inputs (I_0, I_1 and I_2), two recurrent inputs (R_0 and R_1), five outputs (Y_0, Y_1, Y_2, Y_3 and Y_4), weights ($w_0, w_1, w_2, w_3, ..., w_{11}$) and the plastic feedback (PF).

Figure 9 shows the original phenotype having the following genotype:

$$\begin{aligned}\text{Genotype} = & f_0, I_0, w_0, I_1, w_1, I_2, w_2, f_1, I_3, w_5, R_1, w_4, I_2, w_6, \\ & f_2, I_3, w_7, R_1, w_4, R_0, w_8, f_3, I_3, w_{10}, I_4, w_{11}, I_0, w_9, I_3, I_1, I_4, I_5, I_6\end{aligned}$$

Figure 10 illustrates the change in function as a result of change in function gene as highlighted in the genotype below:

$$\begin{aligned}\text{Genotype} = & f_0, I_0, w_0, I_1, w_1, I_2, w_2, f_1, I_3, w_5, R_1, w_4, I_2, w_6, \\ & f_0, I_3, w_7, R_1, w_4, R_0, w_8, f_3, I_3, w_{10}, I_4, w_{11}, I_0, w_9, I_3, I_1, I_4, I_5, I_6\end{aligned}$$

Figure 11 shows the change in the output connectivity at runtime respectively with corresponding change in gene highlighted below:

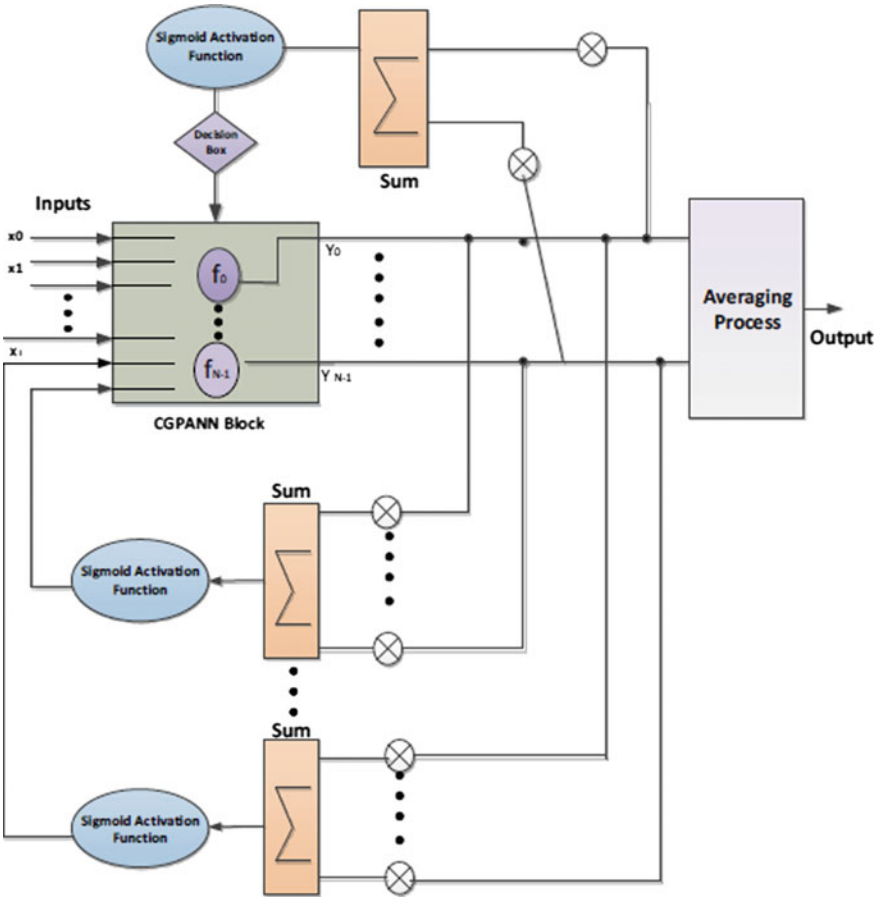

Fig. 8 Structural view of PRCGPANN

$$\text{Genotype} = f_0, I_0, w_0, I_1, w_1, I_2, w_2, f_1, I_3, w_5, R_1, w_4, I_2, w_6,$$
$$f_0, I_3, w_7, R_1, w_4, R_0, w_8, f_3, I_3, w_{10}, I_4, w_{11}, I_0, w_9, I_0, I_1, I_4, I_5, I_6$$

The various cases of genotype and phenotype diagrams above demonstrate the possible run time modification to the recurrent network, thus adding not only the signal feedback but also structural feedback through modification in the system architecture and topology at runtime. This is a very interesting system, because it can transform to feedforward and feedback structure from time to time during the processing, thus giving a unique ability to the system. This architecture is yet to be explored on various applications.

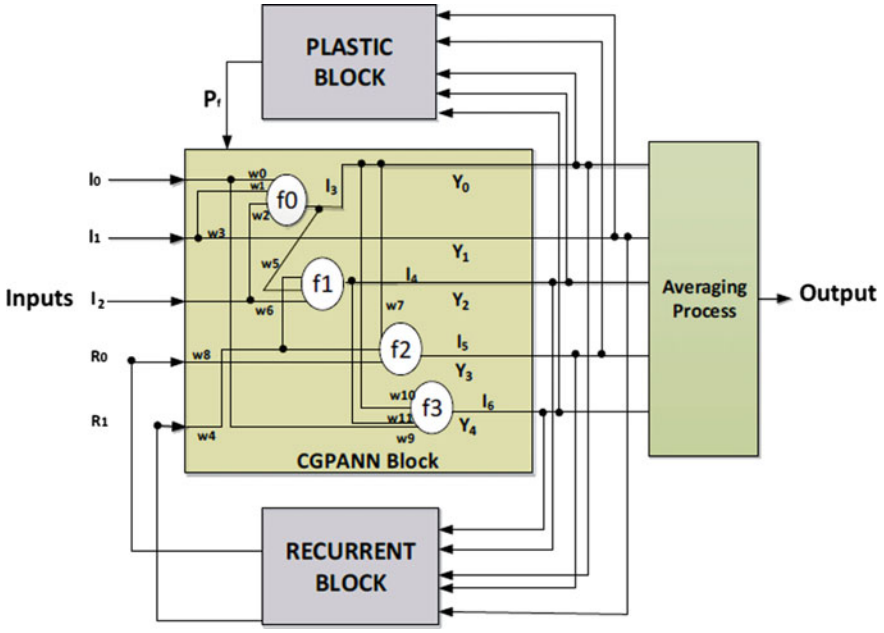

Fig. 9 Original PRCGPANN phenotype

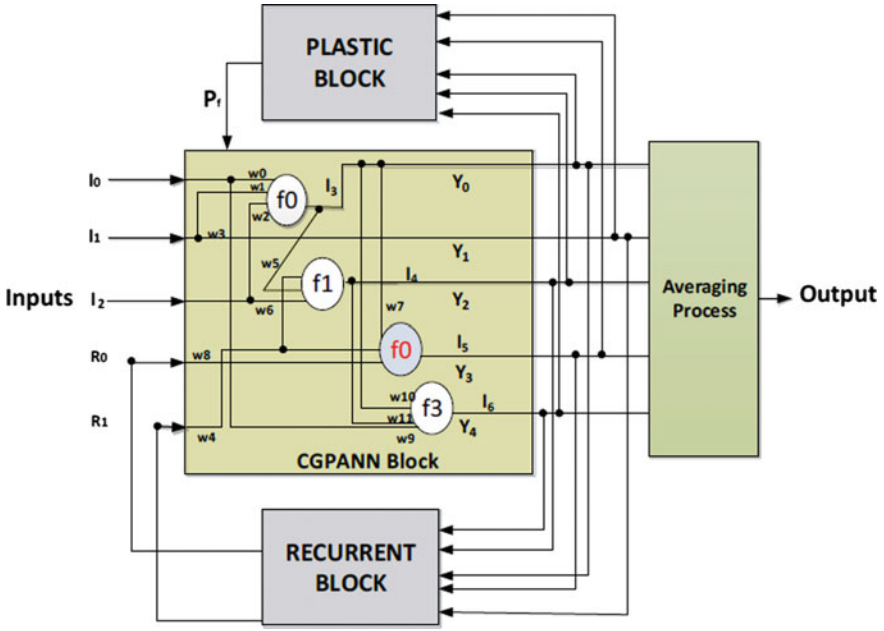

Fig. 10 Mutation in the function of the network

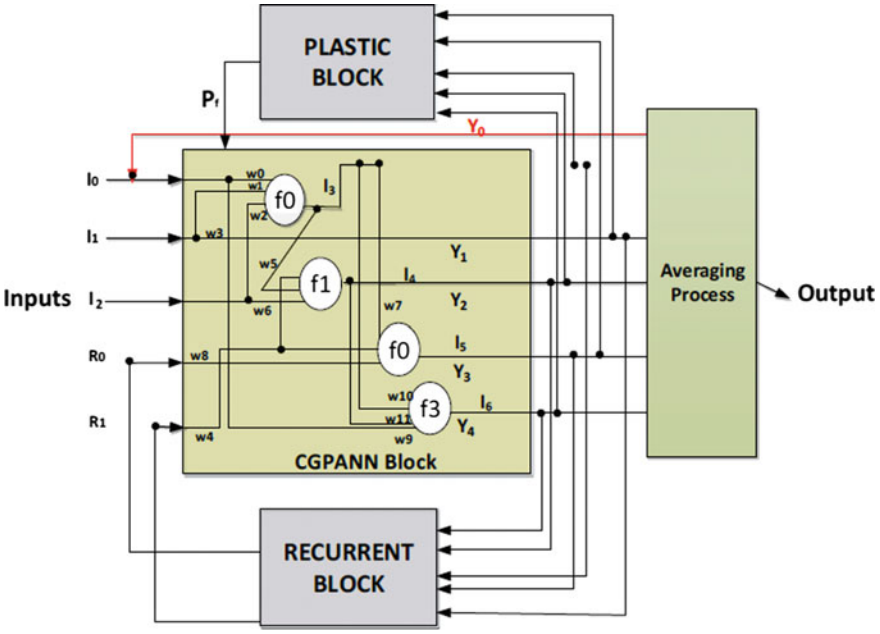

Fig. 11 Mutation in the output gene

4 Concluding Remarks

This chapter provides a detailed overview of how CGP is used to evolve artificial neural network by finding the proper set of weights and topology for the network. CGP based ANN provides an ideal platform to all Markovian and non-Markovian, Linear and non-linear problems that are static or dynamic/plastic. They can help finding the unknown mathematical model for the problem at hand. The CGPANN model not only helps in selection of topology and optimum weights for ANNs, but also helps in identifying the best possible features to be selected amongst many provided to the network and ignoring the unwanted noise. Various models of CGPANNs are tested in diverse fields of application for its speed of learning, robustness, and accuracy. Comparison with other algorithms on same set of problems show encouraging results.

References

1. Ali, J., Khan, G.M., Mahmud, S.A.: Enhancing growth curve approach using CGPANN for predicting the sustainability of new food products. In: IFIP International Conference on Artificial Intelligence Applications and Innovations, pp. 286–297. Springer (2014)

2. Arbab, M.A., Khan, G.M., Sahibzada, A.M.: Cardiac arrhythmia classification using cartesian genetic programming evolved artificial neural network. Exp. Clin. Cardiol. **20**(9) (2014)
3. Bidlo, M.: Evolutionary design of generic combinational multipliers using development. In: International Conference on Evolvable Systems, pp. 77–88. Springer (2007)
4. Carpenter, G.A., Grossberg, S.: The art of adaptive pattern recognition by a self-organizing neural network. Computer **21**(3), 77–88 (1988)
5. Fekiač, J., Zelinka, I., Burguillo, J.C.: A review of methods for encoding neural network topologies in evolutionary computation. In: Proceedings of 25th European Conference on Modeling and Simulation ECMS 2011, pp. 410–416 (2011)
6. Gomez, F., Schmidhuber, J., Miikkulainen, R.: Accelerated neural evolution through cooperatively coevolved synapses. J. Mach. Learn. Res. **9**(May), 937–965 (2008)
7. Gomez, F.J., Miikkulainen, R.: Solving non-markovian control tasks with neuroevolution. IJCAI **99**, 1356–1361 (1999)
8. Guo, H., Nandi, A.K.: Breast cancer diagnosis using genetic programming generated feature. Pattern Recognit. **39**(5), 980–987 (2006)
9. Haider, A., Hanif, M.N.: Inflation forecasting in Pakistan using artificial neural networks. Pak. Econ. Soc. Rev. 123–138 (2009)
10. Iranpour, M., Almassi, S., Analoui, M.: Breast cancer detection from fna using svm and rbf classifier. In: 1st Joint Congress on Fuzzy and Intelligent Systems (2007)
11. Jordan, M.I.: Attractor dynamics and parallellism in a connectionist sequential machine. In: Proceedings of the 8th Confererence of the Cognitive Science Society, pp. 531–546. Lawrence Erlbaum Associates (1986)
12. Kamruzzaman, J., Sarker, R.A.: Forecasting of currency exchange rates using ANN: a case study. In: Neural Networks and Signal Processing, 2003. Proceedings of the 2003 International Conference on, vol. 1, pp. 793–797. IEEE (2003, December)
13. Khan, G.M., Ali, J., Mahmud, S.A.: Wind power forecastingan application of machine learning in renewable energy. In: 2014 International Joint Conference on Neural Networks (IJCNN), pp. 1130–1137. IEEE (2014)
14. Khan, G.M., Khattak, A.R., Zafari, F., Mahmud, S.A.: Electrical load forecasting using fast learning recurrent neural networks. In: The 2013 International Joint Conference on Neural Networks (IJCNN), pp. 1–6. IEEE (2013)
15. Khan, G.M., Nayab, D., Mahmud, S.A., Zafar, H.: Evolving dynamic forecasting model for foreign currency exchange rates using plastic neural networks. In: 2013 12th International Conference on Machine Learning and Applications (ICMLA), vol. 2, pp. 15–20. IEEE (2013)
16. Khan, G.M., Ullah, F., Mahmud, S.A.: MPEG-4 internet traffic estimation using recurrent CGPANN. In: International Conference on Engineering Applications of Neural Networks, pp. 22–31. Springer (2013)
17. Khan, G.M., Zafari, F.: Dynamic feedback neuro-evolutionary networks for forecasting the highly fluctuating electrical loads. Genet. Program. Evolvable Mach. **17**(4), 391–408 (2016)
18. Khan, G.M., Zafari, F., Mahmud, S.A.: Very short term load forecasting using cartesian genetic programming evolved recurrent neural networks (CGPRNN). In: 2013 12th International Conference on Machine Learning and Applications (ICMLA), vol. 2, pp. 152–155. IEEE (2013)
19. Khan, M.M., Ahmad, A.M., Khan, G.M., Miller, J.F.: Fast learning neural networks using cartesian genetic programming. Neurocomputing **121**, 274–289 (2013)
20. Khan, M.M., Khan, G.M., Miller, J.F.: Developmental plasticity in cartesian genetic programming artificial neural networks. In: Proceedings of the 8th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2011), pp. 449–458. SciTePress (2011)
21. Kryuchin, O.V., Arzamastsev, A.A., Troitzsch, K.G.: The prediction of currency exchange rates using artificial neural networks. Exch. Organ. Behav. Teach. J. **4** (2007)
22. Liu, K., Subbarayan, S., Shoults, R., Manry, M., Kwan, C., Lewis, F., Naccarino, J.: Comparison of very short-term load forecasting techniques. IEEE Trans. Power Syst. **11**(2), 877–882 (1996)
23. Moriarty, D.E.: Symbiotic evolution of neural networks in sequential decision tasks. Ph.D. thesis, University of Texas at Austin USA (1997)

24. Nayab, D., Khan, G.M., Mahmud, S.A.: Prediction of foreign currency exchange rates using cgpann. In: International Conference on Engineering Applications of Neural Networks, pp. 91–101. Springer (2013)
25. Pan, Z., Nie, L.: Evolving both the topology and weights of neural networks. Parallel Algorithms Appl. **9**(3–4), 299–307 (1996)
26. Papadrakakis, M., Papadopoulos, V., Lagaros, N.D.: Structural reliability analyis of elastic-plastic structures using neural networks and monte carlo simulation. Comput. Methods Appl. Mech. Eng. **136**(1–2), 145–163 (1996)
27. Philip, A.A., Taofiki, A.A., Bidemi, A.A.: Artificial neural network model for forecasting foreign exchange rate. World Comput. Sci. Inf. Technol. J. (WCSIT) **1** (3) 110–118 (2011)
28. Pujol, J.C.F., Poli, R.: Evolving the topology and the weights of neural networks using a dual representation. Appl. Intell. **8**(1), 73–84 (1998)
29. Refenes, A.N., Azema-Barac, M., Chen, L., Karoussos, S.: Currency exchange rate prediction and neural network design strategies. Neural Comput. Appl. **1**(1), 46–58 (1993)
30. Rehman, M., Ali, J., Khan, G.M., Mahmud, S.A.: Extracting trends ensembles in solar irradiance for green energy generation using neuro-evolution. In: IFIP International Conference on Artificial Intelligence Applications and Innovations, pp. 456–465. Springer (2014)
31. Rehman, M., Khan, G.M., Mahmud, S.A.: Foreign currency exchange rates prediction using cgp and recurrent neural network. IERI Procedia **10**, 239–244 (2014)
32. Sadeghi, B.: A bp-neural network predictor model for plastic injection molding process. J. Mater. Process. Technol. **103**(3), 411–416 (2000)
33. Sarangi, P.P., Sahu, A., Panda, M.: A hybrid differential evolution and back-propagation algorithm for feedforward neural network training. Int. J. Comput. Appl. **84**(14) (2013)
34. Singh, D., Singh, S.: A self-selecting neural network for short-term load forecasting. Electr. Power Compon. Syst. **29**(2), 117–130 (2001)
35. Stanley, K.O., Miikkulainen, R.: Efficient reinforcement learning through evolving neural network topologies. In: Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation, pp. 569–577. Morgan Kaufmann Publishers Inc. (2002)
36. Wieland, A.P.: Evolving neural network controllers for unstable systems. In: IJCNN-91-Seattle International Joint Conference on Neural Networks, 1991, vol. 2, pp. 667–673. IEEE (1991)