Hindawi Publishing Corporation
Advances in Artificial Neural Systems
Volume 2011, Article ID 532785, 9 pages
doi:10.1155/2011/532785

Research Article

Genetic Algorithm-Based Artificial Neural Network for Voltage Stability Assessment

Garima Singh and Laxmi Srivastava

Electrical Engineering Department, Madhav Institute of Technology and Science, Gwalior 474 005, India

Correspondence should be addressed to Laxmi Srivastava, srivastaval@hotmail.com

Received 3 May 2011; Accepted 16 June 2011

Academic Editor: Ping Feng Pai

With the emerging trend of restructuring in the electric power industry, many transmission lines have been forced to operate at almost their full capacities worldwide. Due to this, more incidents of voltage instability and collapse are being observed throughout the world leading to major system breakdowns. To avoid these undesirable incidents, a fast and accurate estimation of voltage stability margin is required. In this paper, genetic algorithm based back propagation neural network (GABPNN) has been proposed for voltage stability margin estimation which is an indication of the power system's proximity to voltage collapse. The proposed approach utilizes a hybrid algorithm that integrates genetic algorithm and the back propagation neural network. The proposed algorithm aims to combine the capacity of GAs in avoiding local minima and at the same time fast execution of the BP algorithm. Input features for GABPNN are selected on the basis of angular distance-based clustering technique. The performance of the proposed GABPNN approach has been compared with the most commonly used gradient based BP neural network by estimating the voltage stability margin at different loading conditions in 6-bus and IEEE 30-bus system. GA based neural network learns faster, at the same time it provides more accurate voltage stability margin estimation as compared to that based on BP algorithm. It is found to be suitable for online applications in energy management systems.

1. Introduction

Voltage stability is concerned with the ability of the power system to maintain acceptable voltages at all the system buses under normal conditions as well as after being subjected to a disturbance. Thus, the analysis of voltage stability deals with finding the voltage levels at all buses in the system under different loading conditions to ascertain the stability limit and margin. A power system enters a state of voltage instability when a disturbance, increase in load demand or change in system conditions causes a progressive and uncontrollable decline of bus voltages.

The main factor causing voltage instability is the inability of the power system to meet the reactive power demand. In most of the cases, voltage profiles show no abnormality prior to undergoing voltage collapse because of the load variation. Voltage stability margin (VSM) is a static voltage stability index which is used to quantify how "close" a particular operating point is to the point of voltage collapse [1]. Thus, VSM may be used to estimate the steady-state voltage stability limit of a power system. Knowledge of the voltage stability margin is of vital importance to utilities in order to operate their system with appropriate security and reliability

During the last few years, several methodologies for detecting the voltage collapse points (saddle-node bifurcations) in power systems using steady-state analysis techniques have been modified and applied for the determination of analyzing voltage stability of power systems for example PV and QV curves, sensitivity-based indices [2] and continuation power flow methods [3, 4]. Other methods, such as bifurcation theory [5], energy function [6], singular value decomposition [7], and so forth, have been also reported in the literature.

These analytical methods involve considerable computational effort and require significantly large computational time and, hence, cannot be used directly for online monitoring and initiation of preventive control actions to enhance system voltage stability. For online applications, there is a need for quick detection of the potentially dangerous

situations of voltage instability so that necessary actions may be taken to avoid the occurrence of voltage collapse in a power system.

Recently, artificial neural networks (ANNs) have been proposed for voltage stability evaluation [1, 8–16] as they have the ability to properly classify a highly nonlinear relationship, and, once trained, they can classify new data much faster than it would be possible by solving the model analytically. However, most of the published work in the area of voltage stability employed either multilayer perceptron networks trained by back propagation algorithm [8–14]. In reference [13], the energy function-based voltage stability indicator is predicted using the multilayer perception (MLP) with a second-order learning rule and the radial basis function (RBF) network. The input to the neural network consists of real and reactive power injections at all load buses in the system for a particular loading condition, while the output of the network is the energy margin. In [14], an approach based on artificial feed-forward neural network (FFNN) is presented for assessing power system voltage stability. The approach uses real and reactive power, as well as voltage vectors for generators and load buses to train the neural net (NN). The input properties of the NN are generated from offline training data with various simulated loading conditions using a conventional voltage stability algorithm-based on the L-index. In reference [1] parallel self-organizing hierarchical neural network is developed for assessing power system voltage stability while in [15, 16], Kohonen's self-organizing feature map (SOFM) has been proposed for quantifying stability margins.

In typical power systems, there are voluminous amount of input data. Then, the success of ANN applications also depends on the systematic approach of selecting highly important features which will result in a compact and efficient ANN. Different feature reduction methods for voltage stability assessment are compared in [17]. Feature reduction is crucial for the success of ANN application, although each has its own merit and demerit. Feature selection based on clustering technique can identify important parameters directly measurable from the power system.

Voltage instability is, in general, caused by either of two types of system disturbances: increase in load demand and contingencies. In the present paper, voltage instability due to increase in load demand is considered. A genetic algorithm-based back propagation neural network [18–23] has been proposed for voltage stability margin estimation which evaluates system stability from static viewpoint. Input features for GABPNN are selected by applying angular distance-based clustering technique in the real and reactive loads [24, 25].

These conventional methods of voltage stability assessment are computationally intensive and data sensitive. On the other hand, artificial neural network-based approach is fast and provides result even with partially missing/noisy data. Back propagation (BP) searches on the error surface by means of the gradient descent technique in order to minimize the error. It is therefore likely to get struck in a local minimum [18]. On the other hand in GABPNN, there exist genetic algorithms (GA) which are adaptive search and optimization algorithms that mimic the principles of natural genetics. Genetic algorithms are quite different from traditional search and optimization techniques used in engineering design problems but at the same time exhibit simplicity, ease of operation, minimal requirements, and global perspective.

2. Genetic Algorithm-Based BP Neural Network

The idea to hybridize the two approaches, namely, GA and BPN follows naturally. Rajasekaran and Pai [20] used GAs to guide back propagation network in finding the necessary connections instead of full connections in order to enhance the speed of training. The general schematic of the genetic algorithm (GA) is considered to be a stochastic heuristic (or metaheuristic) method. Genetic algorithms are inspired by adaptive and evolutionary mechanisms of live organisms.

Genetic algorithm is an adaptive search technique used for solving mathematical problems and engineering optimization problems that emulates Darwin's evolutionary theory that is fittest is likely to survive. Genetic algorithm attempts to find a good (or best) solution to the problem by genetically breeding a population of individuals over a series of generations. In genetic algorithm, each individual in the population represents a candidate solution to the given problem. The GA transforms a population (set) of individuals, each with an associated fitness value, into a new generation of the population using reproduction, crossover and mutation. Core of the GA is genetic recombination of strings. Generally, a population of strings is randomly generated at the beginning of the process.

An important characteristic of GA is that global feature of search is related to the diversity of the initial population: the more diverse the population, the more global the search. From the initial population, selection strategy based on fitness proportion is adopted to select individuals in current population. Higher selective pressure often leads to the loss of diversity in the population, which causes premature convergence but at the same time improves convergence speed. Therefore, a balance is required between population diversity and convergence speed for obtaining the good performance of GA. Then reproduction, cross-over, and mutation operators are randomly applied to produce next generation population until genetic stopping condition is satisfied.

When the GA is correctly implemented for solving any problem, the population evolves over successive iterations with the fitness value increasing towards global optimum. Several features of GAs like no dependency on gradient information, less likely to be trapped in local minima, ability to deal with the problems where no explicit/exact objective function is available, ability to deal with the concave objective function-based optimization problems, and make them much more robust than many other search algorithms. Moreover GAs are much superior to conventional search and optimization techniques in high-dimensional problem space due to their inherent parallelism and directed stochastic search implemented by recombination operators.

Table 1: Genetic algorithm operations.

(A) Generate randomly an initial population of individuals.
(B) Carry out the following substeps iteratively for each generation until a termination condition is fulfilled.
(i) Evaluate fitness of each individual to check its ability to solve the specific problem and save the best individual of all preceding population.
(ii) Select pair of individuals to be parents for reproduction on the basis of their fitness.
(iii) Generate offsprings from parents by implementing genetic search operators such as cross-over/mutation. Add them to the population.

GA operates[18] through a simple cycle of three stages, which are described in Table 1.

As shown in Table 1, in substep (ii), *selection* is based on fitness, that is, the fitter an individual the greater the chance for this individual to get selected for reproduction and contribute offspring for the next generation. *Cross-over* operator takes two chromosomes and swaps part of their genetic information to produce new chromosomes. *Mutation* is implemented by occasionally altering a random bit in a string before the offsprings are inserted into the new population. The performance of GA is achieved by having a balanced combination of three control parameters. These control parameters are crossover probability, mutation probability, and population size. Some important observations related to GA are the following:

(i) Increasing the crossover probability increases the recombination of building blocks. But it also increases the disruption of good strings.

(ii) Increasing the mutation probability tends to transform the genetic search into random search, but it also helps reintroduce lost genetic material.

(iii) Increasing the population size increases the diversity and reduces the probability of premature convergence to a local optimum, but it also increases the time required for the population to converge to the optimal region in the search space.

2.1. Weight Optimization Using GA for ANN Training. Artificial neural networks and genetic algorithms are both abstractions of natural processes. They are formulated into a computational model so that the learning power of neural networks and adaptive capabilities of evolutionary processes can be combined [18]. Genetic algorithms can help to determine optimized neural network interconnection weights, as well as, to provide faster mechanism for training of the neural network. Training a given neural network generally means to determine an optimal set of connection weights. This is formulated as the minimization of some network error functions, over the training data set, by iteratively adjusting the weights. The mean square error between the target and actual output averaged over all output nodes serves as a good estimate of the fitness of the network configuration corresponding to the current input. Conventionally a back-propagation neural network (BPNN) updates its weights through a gradient descent technique with backward error propagation. This gradient search technique sometimes gets stuck into local minima. Gas, on the other hand, though not guaranteed to find global optimum solution, have been found to be good at finding "acceptably good" solutions "acceptably quickly" [18].

Table 2: General framework of GAs for neural network training.

(i) Decode each individual in the current population into a set of connection weights and construct a corresponding ANN with the weights.
(ii) Evaluate the ANN by computing its total mean square error between actual and target outputs.
(iii) Determine fitness of individual as inverse of error. The higher is the error, the lower is the fitness.
(iv) Store the weights for mating pool formation.
(v) Implement search operators such as cross-over/mutation to parents to generate offsprings.
(vi) Calculate fitness for new population.
(vii) Repeat steps (iii) to (vi) until the solution converge.
(viii) Extract optimized weights.

The GA-based weight optimization during training of an ANN follows two steps. The first step is encoding strings for the representation of connection weights. The second step is the evolutionary process simulated by GA, in which search operators have to be implemented in conjunction with the representation scheme. The evolution stops when the population has converged. A population is said to have converged when 95% of the individuals constituting the population share the same fitness value [20]. The whole process for neural network training using a genetic algorithm is shown in Table 2.

Obstacles to the success of GA in evolving the weights for a fixed network structure include the manner in which weights are encoded to the chromosomes and the definition of the "fitness function" that allows the preferential reproduction of good offsprings and prevents premature convergence to a poor solution [18, 20]. Although GA offers an attractive way to optimize ANN weights, it is relatively slow in local fine tuning in comparison to gradient methods. A desirable approach in this case would be to integrate a local gradient search with the GA.

2.2. Weight Extraction. To determine the fitness value for each of the chromosomes, we extract weight from each of chromosomes.

Let $x_1, x_2, \ldots, x_d, \ldots, x_L$ represent a chromosome and $x_{kd+1}, x_{kd+2}, \ldots, x_{(k+1)d}$ represent the kth gene ($k \geq 0$) in the chromosome. The actual weight is given by

$$w_k = \begin{cases} +\dfrac{x_{kd+2}10^{d-2} + x_{kd+3}10^{d-3} + \cdots x_{(k+1)d}}{10^{d-2}}, \\ \qquad\qquad \text{if } 5 \leq x_{kd+1} \leq 9, \\ -\dfrac{x_{kd+2}10^{d-2} + x_{kd+3}10^{d-3} + \cdots x_{(k+1)d}}{10^{d-2}}, \\ \qquad\qquad \text{if } 0 \leq x_{kd+1} \leq 5. \end{cases} \quad (1)$$

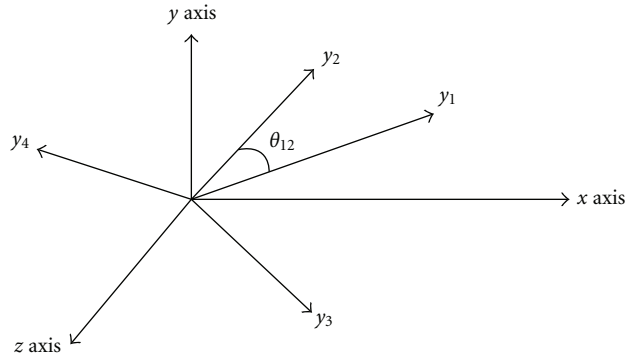

Figure 1: Vectors y shown in 3-dimensional space.

2.3. Feature Selection. In any neural network application, if a large number of input variables are used, the number of interconnection weights will increase and the training of neural network will be extremely slow. To overcome this problem, those variables are selected as input to a neural network that has significant effect on its output, that is, on voltage stability margin. Performance of any neural network mainly depends upon the input features selected for its training. It is essential to reduce the number of inputs to a neural network and to select its optimum number.

To select the input features, angular distance base-clustering method is used. The basic principle for clustering is to group the total N system variables $(S_1, S_2, \ldots, S_N)$ into G clusters such that the variables in a cluster have similar characteristics and then pick out one representative variable from a cluster as the feature for that cluster. This will reduce the number of system variables from N to G. For the purpose of feature selection using the clustering technique, let M training patterns are created.

With the state vector $X'' = [x'_{i1}, x'_{i2}, \ldots, x'_{iN}]$ $(i = 1, 2, \ldots, M)$ at hand, From the matrix X'

$$X' = \begin{bmatrix} x'_{11} & x'_{12} & \ldots & x'_{1N} \\ x'_{21} & x'_{22} & \ldots & x'_{2N} \\ \vdots & \vdots & \ldots & \vdots \\ x'_{M1} & x'_{M2} & \ldots & x'_{MN} \end{bmatrix} \begin{matrix} \longleftarrow \text{ operating point } 1 \\ \longleftarrow \text{ operating point } 2 \\ \\ \longleftarrow \text{ operating point } M \end{matrix} \tag{2}$$

It is observed that row i of matrix X' contains the value for N-system variables $(S_1, S_2, \ldots, S_N)$ at operating point i while column j of matrix X' consists of the S_j variable in the M training patterns. Define the column vector $Y_j = [X'_{lj}, \ldots, X'_{Mj}]^T = [Y'_{lj}, \ldots, Y'_{Mj}]^T$. Then the variables S_j can be clustered based on these vectors Y_j.

Those system variables with similar vector Y will be grouped in a cluster. As shown in Figure 1, two vectors $y1$ and $y2$ which are similar to each other will have a small angle between them. We can, therefore, put a group of these similar vectors in a cluster. To do this, define the cosine value of the angle θ_{jk} between two vector Y_j and Y_k as

$$\cos\left(\varphi_{jk}\right) = \frac{\left(Y_j Y_k\right)}{\left(\left|Y_j\right| |Y_k|\right)}. \tag{3}$$

This cosine value can be used to evaluate the degree of similarity between two vectors. If $\cos(\theta_{jk})$ is greater than a specified threshold ρ_t, the two vectors Y_j and Y_k are regarded as two similar vectors and are put in the same cluster. Details of clustering algorithm are described as follows.

Step 1. Let the system variable S_l belong to cluster 1 and the cluster vector for cluster 1 be equal to column $C_l = [c_{l1}, \ldots, c_{Ml}]^T$ vector Y_l, that is,

$$c_{il} = y_{il} \quad \text{for } l = 1, 2, \ldots, M. \tag{4}$$

Set initial count $j = 0$, $G = 1$.

Step 2. Increase j by one.

Step 3. Compute the cosine value D_g between vector Y_j and C_g $(g = 1, 2, \ldots, G)$

$$D_g = \frac{\left(\sum_{i=1}^{M} y_{ij} c_{ig}\right)}{\left(\sum_{i=1}^{M} y_{ij}^2 \sum_{i=1}^{M} c_{ig}^2\right)^{1/2}}, \quad g = 1, 2, \ldots, G. \tag{5}$$

Let $D_{gm} = \max(D_g)$.

If $D_{gm} < \rho_t$ (a specified threshold), the present vector Y_j is far from any existing cluster vectors, and we have to create a new cluster for system variable S_j.

Step 4. If $D_{gm} > \rho_t$, the present vector Y_j is close to cluster vector C_{gm} of cluster g_m, and system variable S_j should be assigned g_m. Proceed to Step 4.

If vector Y_j has not been presented before, let system variable S_j be grouped in cluster g_m, then update the cluster vector $C_{gm} = [c_{1gm}, c_{2gm}, \ldots, c_{Mgm}]^T$ as follows:

$$c_{igm} = y_{il} + c_{igm} k_{gm} \quad i = 1, 2, \ldots, M, \tag{6}$$

where k_{gm} is the number of system variable in cluster g_m. Go to Step 6. If the vector Y_j has been presented before and variable S_j has been grouped in cluster g_m, go to Step 6.

If vector Y_j has been presented before and variable S_j has been grouped in a different cluster $g_{m'}$, move system variable S_j to cluster g_m and execute the update formulae.

$$\begin{aligned} C_{igm} &= y_{ij} + C_{igm} k_{gm}, \quad i = 1, 2, \ldots, M, \\ C_{igm'} &= -y_{ij} + C_{igm'} k_{gm'}, \quad i = 1, 2, \ldots, M, \end{aligned} \tag{7}$$

where k_{gm} is the number of system variable in cluster vector g_m. In this case, this is a move in cluster elements. Go to Step 6.

Step 5. Create a new cluster g_n for system variable S_j with the cluster vector

$$C_{gm} = \left[c_{1gm}, c_{2gm}, \ldots, c_{Mgm}\right]^T, \tag{8}$$

where $c_{igm} = y_{ij}$, $i = 1, 2, \ldots, M$.

Increase G by one and go to Step 6.

Step 6. If $j = N$, proceed to Step 7; otherwise, go to Step 2.

Step 7. If there is any move in cluster elements in the preceding N iterations, reset j to zero and proceed to Step 2. Otherwise, go to Step 8.

Step 8. For each cluster g, find a system variable S_g whose column vector y_g is closest to the cluster vector C_g, that is,

$$\frac{\left(\sum_{i=1}^{M} y_{gj} c_{ig}\right)}{\left(\sum_{i=1}^{M} y_{gj}^2 \sum_{i=1}^{M} c_{ig}^2\right)^{1/2}} \geq \frac{\left(\sum_{i=1}^{M} y_{ij} c_{ig}\right)}{\left(\sum_{i=1}^{M} y_{ij}^2 \sum_{i=1}^{M} c_{ig}^2\right)^{1/2}}. \quad (9)$$

For any S_j in cluster g, let system variable S_g be the feature for cluster g. Store the system variables and feature variable for each cluster and stop.

2.4. Data Normalization. During training of a neural network, the higher valued input variables may tend to suppress the influence of smaller ones. To overcome this problem, the neural networks are trained with normalized input data, leaving the network to learn weights associated with the connections emanating from these inputs. The raw data are scaled in the range 0.1–0.9 for use by neural networks to minimize the effect of magnitude between input [18]. In case of output variable, if it assumes values close to unity (≥ 1) or (0), it causes difficulty in training as the value unity or zero are practically never realized by the activation or threshold function. A way to overcome this difficulty is to normalize the variables ($\times$) to keep its values between some suitable range (say 0.1 and 0.9). In the present application, each input parameter ($\times$) is normalized as $\times_n$ before being applied to the neural network according to the following equation:

$$x_n = \frac{0.8 \times (x - x_{\min})}{x_{\max} - x_{\min}} + 0.1, \quad (10)$$

where $x_{\max}$ and $x_{\min}$ are the maximum and minimum value of data parameter X. The input data are normalized between 0.9 and 0.1. Similarly, output data (VSM) are normalized between 0.9 and 0.1. During testing phase, the output of the neural network is demoralized to obtain the actual value of voltage stability margin.

3. Solution Algorithm

Sequential steps (flowchart) for developing GABPNN proposed for voltage stability margin estimation are illustrated in Figure 2. The algorithm for VSM estimation has been summarized as follows.

Step 1. Generate a large number of load patterns by perturbing the loads at all the buses in wide range randomly.

Step 2. Normalize input data as selected from the angular distance base-clustering method and the output, that is, voltage stability margin λ_0 to train the GABPNN.

Step 3. Set numbers of generations for genetic optimization of weights.

Step 4. Initialize structure for the neural network, that is, input-hidden-output nodes for determining the number of weights.

Step 5. Set generation count as $g = 0$.

Step 6. Generate randomly the initial population p_g of real coded chromosomes C_g^G, where G is population size.

Step 7. Extract weights for each of the population P_g.

Step 8. Calculate the fitness value for each individual in the population as,

$$\text{Fitness} = \frac{1}{(\text{RMS error})}. \quad (11)$$

Step 9. Get the mating pool ready by replacing worst fit individuals with high-fit individuals.

Step 10. Using cross-over mechanism, reproduce offsprings from the parent chromosomes.

Step 11. Next generation population is achieved. Increase generation count by 1, that is,

$$g = g + 1. \quad (12)$$

Step 12. Check, if $g < G$, then go to Step 7, otherwise go to next step.

Step 13. Training is complete. Extract optimized weights from converged population P_G.

Step 14. Test the developed GABPNN for unseen load patterns.

In the proposed GABPNN, GA performs the weight adaptation for acquiring the minimized error during the training. Before executing certain task, GA requires several parameters to be devised for its proper functioning. Some of them are gene encoding, population initialization, selection, reproduction, fitness evaluation, and so forth. The basic computing elements in GAs are *genes*, which are joined together to form a string of values referred to as a *chromosome*. *Genes* encoding is required to represent weights. To carry out efficient weight optimization in ANN, it is important to have a proper encoding scheme.

There is no definite view in the literature about suitability of encoding schemes for chromosomes. Too complicated scheme provides high flexibility in problem representation but may reduce GA's efficiency due to complicated operations. On the other hand, too simple representation may suffer from slow or premature convergence. This requires a careful selection of an encoding scheme such that GA operations are not compromised but still provides enough flexibility to support dynamic weight adaptation. In the present work, real value coding is adopted for *gene* encoding.

Size of the population of individuals (chromosomes) is generated randomly to start the genetic search procedure. The size of population depends upon number of weights

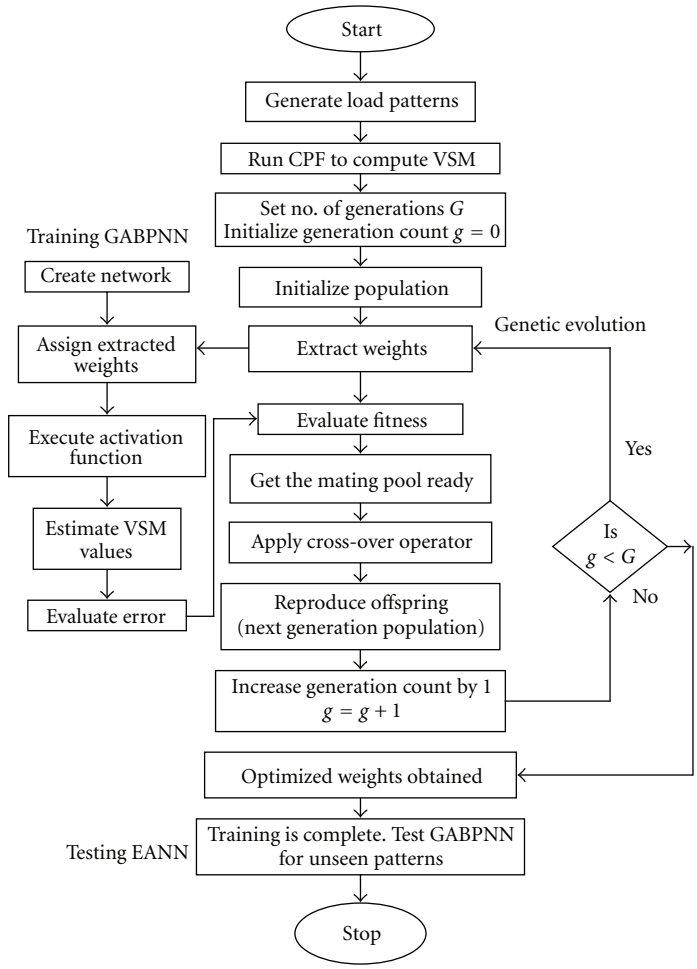

Figure 2: Flow chart for VSM estimation using GABPNN.

to be optimized multiplied by *gene length*. The number of weights to be optimized is determined from neural network configuration. In this work, artificial neural network architecture is assumed to be fixed. If the artificial neural network architecture is *x-y-z,* that is, x input neurons, y hidden neurons and z output neurons, then

$$\text{Number of weights (genes)} = (x + z) * y, \tag{13}$$

$$\text{Population size (chromosome length)} = (x + z) * y * d, \tag{14}$$

where d = *gene length*. Choosing a population size of $(x + z) * y * d$, an initial population of P_i chromosomes is to be randomly generated.

Thus, population size depends upon the number of digits to be used to represent each weight. An appropriate gene length is required for an adequate population size. This way selection of digits to represent a weight is of great importance. Too few digits may result in poor convergence, while a large number of digits per weight will lead to slow convergence due to very long chromosomal string. In the present work, each weight is encoded as a fixed 5-digit string (i.e., $d = 5$), representing weights in the range $[-5,5]$. After deciding the encoding scheme and generating an initial population of *chromosomes*, the program for GABPNN was implemented, as explained above.

Convergence is the progress towards increasing uniformity in fitness values, as each time lowest fitness is replaced with maximum fitness value. Fitness function is taken as inverse of root mean square error (RMS) function. For the converged population, that is, group of individuals comprising minimum RMS, final optimized weights are extracted and decoded. These optimized weights belong to the trained GABPNN, which is ready for testing on unseen load patterns.

4. System Studies

The GA-based back propagation neural network approach has been implemented for voltage stability margin estimation for standard 6-bus system [18] and IEEE 30-bus system [26]. Though the GABPNN-based approach performed well in both the test systems, but due to limited space, the results of only one test system, that is, IEEE 30-bus system

Table 3: Output and error during testing of GABPNN (10-4-1).

TP	Voltage stability margin		Error (p.u.)	Error (% age)	TP	Voltage stability margin		Error (p.u.)	Error (% age)
	Actual	By GABPNN				Actual	By GABPNN		
1	1.45935	1.4304	0.029	1.9849	2	1.62284	1.6169	0.0059	0.3612
3	1.57712	1.5733	0.0038	0.2397	4	1.41375	1.3865	0.0272	1.9244
5	1.64237	1.6341	0.0083	0.5061	6	1.52868	1.5105	0.0182	1.1905
7	1.55574	1.5763	−0.0206	−1.3238	8	1.43946	1.4307	0.0088	0.6145
9	1.5849	1.584	0.0009	0.0561	10	1.47208	1.5002	−0.0281	−1.9118
11	1.41474	1.4116	0.0031	0.2221	12	1.52114	1.5229	−0.0018	−0.1206
13	1.48234	1.5262	−0.0439	−2.9627	14	1.49136	1.4847	0.0067	0.4479
15	1.51966	1.5246	−0.0049	−0.3213	16	1.52114	1.5486	−0.0275	−1.8086
17	1.71256	1.7131	−0.0006	−0.0356	18	1.58577	1.6148	−0.029	−1.8276
19	1.65003	1.6502	−0.0002	−0.011	20	1.59516	1.595	0.0001	0.0048
21	1.45626	1.4538	0.0025	0.1718	22	1.52262	1.5097	0.0129	0.8503
23	1.5655	1.5485	0.017	1.0834	24	1.47739	1.4733	0.0041	0.276
25	1.55784	1.5675	−0.0096	−0.6174	26	1.82254	1.8215	0.0011	0.0619
27	1.43748	1.4152	0.0223	1.5501	28	1.44526	1.4396	0.0057	0.394
29	1.61876	1.5927	0.0261	1.6113	30	1.68389	1.6946	−0.0107	−0.6365

TP: Testing Pattern.

are given in this paper. The IEEE 30-bus system consists of 30 buses and 41 lines. The GABPNN is trained in order to estimate the voltage stability margin λ_o under changing load in normal system operation. For voltage stability margin estimation, 250 load patterns were generated by varying loads randomly at all the buses in the range of 50 to 160% of their base case values and utilizing the corresponding voltage stability margin value as obtained from continuation power flow method (UWPFLOW) [27]. As voltage stability problem usually occurs due to loading condition of a power system, the real and reactive power injections are considered as possible inputs for the GA-based neural network. Input features are selected by applying angular distance-based clustering technique. Using this feature selection method, 8 no. of power injections (P2, P8, P9, P11, P12, Q8, Q11, Q12) were selected with the threshold ρ_t as 0.936. In addition to these features, total real and reactive loads of the power system were selected as input features making the total number of input variable equal to 10.

Out of 250 patterns, 200 patterns are selected randomly for training and the remaining 50 for testing the performance of the trained GABPNN model. During training, it has been found that the number of hidden nodes has affected the convergence rate by increasing or decreasing the complexity of the neural network architecture. Hence, hidden nodes are selected on the "trial and error" basis for obtaining fitness convergence in the minimum number of generations with high convergence rate. In VSM estimation problem, the optimum size of the GABPNN has been found to be 10-4-1. The optimal training for GABPNN was achieved in 15 iterations only.

During testing phase, the 50 testing patterns were tested for evaluating the performance of the trained GABPNN. The testing results of all the 30 patterns are shown in Table 3 and

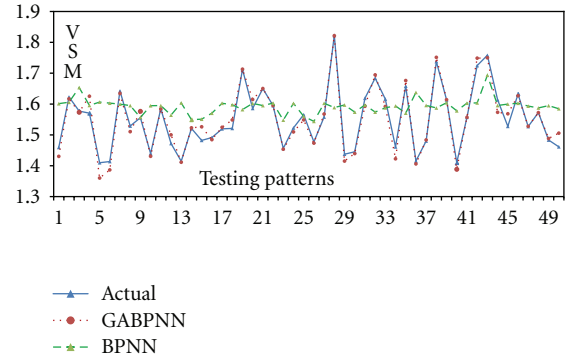

Figure 3: Testing performance of the trained GABPNN and BPNN.

in Figure 3. As can be observed from the table and figure, the trained GABPNN provided good results during testing phase.

4.1. Comparison of GABPNN and BPNN. As the BP neural network is the most popular ANN model and has been implemented in almost every area of engineering, in this paper the performance of GABPNN model has been compared with BPNN model. To compare the effectiveness of the proposed GABPNN approach, a BP model with the same structure that is 10-4-1 was trained for 2500 iterations. The testing results of the trained BPNN are given in Table 4. As can be observed from Table 4, the testing performance of BP model was not satisfactory. This model provided maximum error as 15.71% with rms error of 0.113 pu. On the other hand, the developed GABPNN provided the maximum error of 3.51% and rms error 0.021 pu. However, the training time for both the models was almost same (approximately 13.5 seconds).

Table 4: Output and error during testing of BPNN (10-4-1).

TP	Voltage stability margin		Error (p.u.)	Error (% age)	TP	Voltage stability margin		Error (p.u.)	Error (% age)
	Actual	By BPNN				Actual	By BPNN		
1	1.45935	1.6012	−0.1419	9.72028	2	1.62284	1.60577	0.01707	1.05201
3	1.57712	1.6534	−0.0763	4.83691	4	1.41375	1.59554	−0.1818	12.8586
5	1.64237	1.6064	0.03596	2.18981	6	1.52868	1.60288	−0.0742	4.85421
7	1.55574	1.59965	−0.0439	2.82266	8	1.43946	1.59497	−0.1555	10.8038
9	1.5849	1.55658	0.02833	1.78727	10	1.47208	1.59467	−0.1226	8.32796
11	1.41474	1.595	−0.1803	12.7416	12	1.52114	1.56406	−0.0429	2.82164
13	1.48234	1.60372	−0.1214	8.18847	14	1.49136	1.54934	−0.058	3.88758
15	1.51966	1.55096	−0.0313	2.06016	16	1.52114	1.57032	−0.0492	3.23347
17	1.71256	1.60263	0.10993	6.41887	18	1.58577	1.59772	−0.012	0.75347
19	1.65003	1.58203	0.068	4.1212	20	1.59516	1.60209	−0.0069	0.43449
21	1.45626	1.59506	−0.1388	9.53137	22	1.52262	1.60296	−0.0803	5.27643
23	1.5655	1.54795	0.01755	1.12097	24	1.47739	1.60231	−0.1249	8.45509
25	1.55784	1.56032	−0.0025	0.15901	26	1.82254	1.54385	0.27869	15.2913
27	1.43748	1.60245	−0.165	11.4763	28	1.44526	1.58866	−0.1434	9.92211
29	1.61876	1.59693	0.02184	1.34888	30	1.68389	1.57389	0.11	6.53222

5. Conclusion

In this paper, a hybrid intelligent approach involving genetic algorithm for artificial neural network development has been proposed for voltage stability margin estimation in power system. The fast and accurate estimation of VSM has been considered as an effective way for assessing the stability of a power system from viewpoint of voltage. Implementation of GA makes it possible to achieve effective input-output mapping in artificial neural network with considerable speed-up in its training.

The proposed GABPNN approach sums up the goodness of evolutionary computing and artificial neural networks both. The value of this hybrid approach is that GA requires no gradient information so less susceptible than back-propagation to local variations in the error surface. Another advantageous aspect is that they operate in a population of possible solution candidates in parallel, instead of starting with a single candidate and iteratively operate on it using some sort of heuristics. The proposed approach provides acceptably good generalization ability during testing and found computationally efficient in VSM estimation. Successful application of GABPNN establishes the suitability of the proposed ANN model for online assessment of voltage stability.

Acknowledgments

The authors sincerely acknowledge the financial support provided by Department of Science and Technology (D.S.T), New Delhi, India under Research Grant no F.No.SR/S3/EECE/0064/2009 dated 22-01-10 and Director, Madhav Institute of Technology and Science, Gwalior, India for carrying out this research work.

References

[1] L. Srivastava, S. N. Singh, and J. Sharma, "Estimation of loadability margin using parallel self-organizing hierarchical neural network," *Computers and Electrical Engineering*, vol. 26, no. 2, pp. 151–167, 2000.

[2] P. Kundur, *Power System Stability and Control*, McGraw-Hill, New York, NY, USA, 1994.

[3] V. Ajjarapu and C. Christy, "The continuation power flow: a tool for steady state voltage stability analysis," *IEEE Transactions on Power Systems*, vol. 7, no. 1, pp. 416–423, 1992.

[4] C. A. Canizares and F. L. Alvarado, "Point of collapse and continuation methods for large AC/DC systems," *IEEE Transactions on Power Systems*, vol. 8, no. 1, pp. 1–8, 1993.

[5] C. A. Canizares, "On bifurcations, voltage collapse and load modeling," *IEEE Transactions on Power Systems*, vol. 10, no. 1, pp. 512–522, 1995.

[6] T. J. Overbye and C. L. DeMarco, "Improved techniques for power system voltage stability assessment using energy methods," *IEEE Transactions on Power Systems*, vol. 6, no. 4, pp. 1446–1452, 1991.

[7] P. A. Löf, T. Smed, G. Andersson, and D. J. Hill, "Fast calculation of a voltage stability index," *IEEE Transactions on Power Systems*, vol. 7, no. 1, pp. 54–64, 1992.

[8] B. Jeyasurya, "Artificial neural networks for power system steady-state voltage instability evaluation," *Electric Power Systems Research*, vol. 29, no. 2, pp. 85–90, 1994.

[9] D. Salatino, R. Sbrizzai, M. Trovato, and M. La Scala, "Online voltage stability assessment of load centers by using neural networks," *Electric Power Systems Research*, vol. 32, no. 3, pp. 165–173, 1995.

[10] A. A. El-Keib and X. Ma, "Application of artificial neural networks in voltage stability assessment," *IEEE Transactions on Power Systems*, vol. 10, no. 4, pp. 1890–1896, 1995.

[11] H. P. Schmidt, "Application of artificial neural networks to the dynamic analysis of the voltage stability problem," *IEE Proceedings: Generation, Transmission & Distribution*, vol. 144, pp. 371–376, 1997.

[12] A. Mohamed and G. B. Jasmon, "Neural network approach to dynamic voltage stability prediction," *Electric Machines and Power Systems*, vol. 25, no. 5, pp. 509–523, 1996.

[13] V. R. Dinavahi and S. C. Srivastava, "ANN based voltage stability margin prediction," in *Proceedings of the IEEE Power Engineering Society Summer Meeting*, vol. 2, pp. 1275–1279, Vancouver, Canada, July 2001.

[14] S. Kamalasadan, D. Thukaram, and A. K. Srivastava, "A new intelligent algorithm for online voltage stability assessment and monitoring," *International Journal of Electrical Power & Energy Systems*, vol. 31, no. 2-3, pp. 100–110, 2009.

[15] S. Chauhan and M. P. Dave, "Kohonen neural network classifier for voltage collapse margin estimation," *Electric Machines and Power Systems*, vol. 25, no. 6, pp. 607–619, 1996.

[16] Y. H. Song, H. B. Wan, and A. T. Johns, "Kohonen neural network based approach to voltage weak buses/areas identification," *IEE Proceedings: Generation, Transmission & Distribution*, vol. 144, pp. 340–344, 1997.

[17] W. Nakawiro and I. Erlich, "Online voltage stability monitoring using artificial neural network," in *Proceedings of the 3rd International Conference on Deregulation and Restructuring and Power Technologies (DRPT '08)*, pp. 941–947, Nanjing, China, April 2008.

[18] S. N. Pandey, S. Tapaswi, and L. Srivastava, "Integrated evolutionary neural network approach with distributed computing for congestion management," *Applied Soft Computing Journal*, vol. 10, no. 1, pp. 251–260, 2010.

[19] P. P. Palmes, T. Hayasaka, and S. Usui, "Mutation-based genetic neural network," *IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 587–600, 2005.

[20] S. Rajasekaran and G. A. V. Pai, *Neural Networks, Fuzzy Logic and Genetic Algorithms—Synthesis and Applications*, Prentice-Hall Press, New Delhi, India, 2006.

[21] P. Ramasubramanian and A. Kannan, "A genetic-algorithm based neural network short-term forecasting framework for database intrusion prediction system," *Soft Computing*, vol. 10, no. 8, pp. 699–714, 2006.

[22] S. K. Oh and W. Pedrycz, "Multi-layer self-organizing polynomial neural networks and their development with the use of genetic algorithms," *Journal of the Franklin Institute*, vol. 343, no. 2, pp. 125–136, 2006.

[23] S. K. Oh, W. Pedrycz, and S. B. Roh, "Genetically optimized hybrid fuzzy set-based polynomial neural networks," *Journal of the Franklin Institute*, vol. 348, pp. 415–425, 2011.

[24] Y.-Y. Hsu, C.-R. Chen, and C.-C. Su, "Analysis of electromechanical modes using an artificial neural network," *IEE Proceedings: Generation, Transmission & Distribution*, vol. 141, no. 3, pp. 198–204, 1994.

[25] S. Sharma and L. Srivastava, "Prediction of transmission line overloading using intelligent technique," *Applied Soft Computing Journal*, vol. 8, no. 1, pp. 626–633, 2008.

[26] L. L. Freris and A. M. Sasson, "Investigation of the load-flow problem," *Proceedings of the Institution of Electrical Engineers*, vol. 115, no. 10, pp. 1459–1470, 1968.

[27] C. A. Canizares and F. L. Alvarado, "UWPFLOW Program," University of Waterloo, 2000, http://www.power.uwaterloo.ca/.