Available online at www.sciencedirect.com
SCIENCE DIRECT®
Advanced Engineering Informatics 19 (2005) 255–262
ADVANCED ENGINEERING INFORMATICS
www.elsevier.com/locate/aei

A hybrid intelligent genetic algorithm

A.A. Javadi*, R. Farmani, T.P. Tan

Department of Engineering, School of Engineering, Computer Science and Mathematics, University of Exeter, Exeter, Devon EX4 4QF, UK

Abstract

Application of genetic algorithms to optimization of complex problems can lead to a substantial computational effort as a result of the repeated evaluation of the objective function(s) and the population-based nature of the search. This is often the case where the objective function evaluation is costly, for example, when the value is obtained following computationally expensive system simulations. Sometimes a substantially large number of generations might be required to find optimum value of the objective function. Furthermore, in some cases, genetic algorithm can face convergence problems. In this paper, a hybrid optimization algorithm is presented which is based on a combination of the neural network and the genetic algorithm. In the proposed algorithm, a back-propagation neural network is used to improve the convergence of the genetic algorithm in search for global optimum. The efficiency of the proposed computational methodology is illustrated by application to a number of test cases. The results show that, in the proposed hybrid method, the integration of the neural network in the genetic algorithm procedure can accelerate the convergence of the genetic algorithm significantly and improve the quality of solution.
© 2005 Elsevier Ltd. All rights reserved.

Keywords: Genetic algorithm; Neural network; Hybrid; Optimization

1. Introduction

During the past few decades, genetic algorithms have received a lot of attention regarding their potential as global optimization techniques for complex optimization problems. They are derivative-free stochastic optimization methods based on the concept of natural selection and evolution processes. Their popularity can be attributed to their freedom from dependence on functional derivatives, applicability to both continuous and discrete optimization problems as well as their stochastic nature and reduced likelihood of getting trapped in local optimum which inevitably are present in many practical optimization problems [4,9].

Application of genetic algorithms to optimization of complex problems can lead to a substantial computational effort as a result of the repeated evaluation of the objective function(s) and the population-based nature of the search. This is often the case where the objective function evaluation is costly, for example, when the value is obtained following computationally expensive system simulations. Sometimes a substantially large number of generations might be required to find the optimum value of the objective function. Furthermore, in some cases, genetic algorithm can face convergence problems. This may occur if the crossover operation and selection method are too effective and so, they drive the GA to create a population of individuals that are almost exactly the same. When the population consists of similar individuals, the likelihood of finding new solutions typically decreases.

Neural network is a computational mechanism the structure of which essentially mimics the knowledge acquisition, information processing and organizational skills of a human brain. Neural networks have the capability to learn complex nonlinear relationships and associations in large volumes of data, enabling the analysis of a wide range of pattern recognition problems [10–12].

In recent years, some attempts have been made to combine the merits of genetic algorithms and neural networks in the context of hybrid methods. In majority of the developed hybrid algorithms the focus has been on using genetic algorithms to improve learning of neural networks. For example Adeli and Hung [2] employed a genetic algorithm to perform global search and to seek good starting weight vector for the neural network learning algorithm aiming to improve the convergence speed of the algorithm.

* Corresponding author. Tel.: +44 1392 263640; fax: +44 1392 217965.
E-mail address: a.a.javadi@exeter.ac.uk (A.A. Javadi).

1474-0346/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.aei.2005.07.003

Abu-Alola and Gough [1] used a genetic algorithm to improve learning of a multilayer recurrent neural network.

In this paper, a hybrid neural network-based genetic algorithm is presented which uses neural network to improve solution quality and convergence speed of genetic algorithm. In the proposed algorithm, a back-propagation neural network is trained with the GA results and the trained network is then used to guide the GA in its search for a better optimum value in the subsequent generations. The algorithm has been applied to a number of test functions and has shown to be very efficient in reducing the number of generations required for convergence of GA to optimum as well as attaining significant improvement in solution quality.

2. Genetic algorithms

Genetic algorithms (GAs) as efficient algorithms for solution of optimization problems have been shown to be effective at exploring a large and complex search space in an adaptive way guided by the equivalent biological evolution mechanisms of reproduction, crossover and mutation. They are random search algorithms which have been derived based on the 'Darwin's theory of survival of the fittest' [4–9].

GAs operate on a population of trial solutions that are initially generated at random. The GA seeks to maximize the fitness of the population by selecting the fittest individuals from the population and using their 'genetic' information in 'mating' operations to create a new population of solutions. Many variations of the fundamental GA have been developed, but the algorithm implemented here is derived from the simple GA described by Goldberg [8].

Genetic algorithms have many advantages over the traditional optimization methods. In particular, GAs do not require function derivatives and work on function evaluations alone; they have a better possibility of locating the global optimum because they search a population of points rather than a single point and they allow for consideration of design spaces consisting of a mix of continuous and discrete variables. In addition, GAs provide the decision-makers with a set of acceptable optimal solutions (rather than a single solution) from which they can select the most appropriate one. The probabilistic nature of GA helps to avoid convergence to false optima.

One of the main disadvantages of genetic algorithm techniques is that, although as global optimization techniques, they have good initial convergence characteristics, but they may slow down considerably once the region of optimal solutions has been identified.

Elitism is commonly used in order to improve the convergence of genetic algorithm. It has been well established that elitist evolutionary algorithms (EAs) have better convergence characteristics than non-elitist EAs [5,6]. Generally, elitism preserves the current best solution(s) and transfers them to subsequent generations. In this work, elitism is implemented by simply carrying the best solution of the population to the next generation. In addition, to further improve the convergence of the genetic algorithm, a neural network, trained with the population of individuals and their fitness values, is used to find even a better solution than the current best solution and transfer it to the next generation as well. In this way the efficiency of genetic algorithm is enhanced considerably through the development of a hybrid method, which combines the GA method with neural network. By combining the GA method with neural network, the advantages of both methods are exploited to produce a hybrid optimization method which is both robust and fast. The form of the genetic algorithm used in this study can be summarized as being a simple binary coded GA with roulette wheel selection, uniform crossover and elitist replacement strategy.

3. Neural network

A neural network is an information processing technique developed by inspiration by the way biological nervous systems, such as the brain, process information. Neural networks are composed of a large number of highly interconnected processing elements, or neurons, usually arranged in layers. These layers generally include an input layer, a number of hidden layers and an output layer. Signals that are generated from the input propagate through the network on a layer-by-layer basis in the forward direction. Neurons in hidden layers are used to find associations within the input data and extract patterns than can provide meaningful outputs. A neural network system uses the human-like technique of learning by example to resolve problems. Just as in biological systems, learning involves adjustments to the synaptic connections that exist between the neurons. The output of each neuron, responding to a particular combination of inputs, has an influence (or weight) on the overall output. Weighting is controlled by the level of activation of each neurone, and the strength of connection between individual neurons. Patterns of activation and interconnection are adjusted to achieve the desired output from the training data. Corrections are based on the difference between actual and desired output, which is computed for each training cycle. If average error is within a prescribed tolerance the training is stopped, the weights are locked in and the network is ready to use [3,15].

Generally, there are two main types of learning: supervised and unsupervised. In a supervised learning, the external environment also provides a desired output for each one of the training vectors whereas in unsupervised learning, the external environment does not provide the desired network output. In the present work, a supervised learning was adopted as the desired outputs are available for the input vectors from the GA runs.

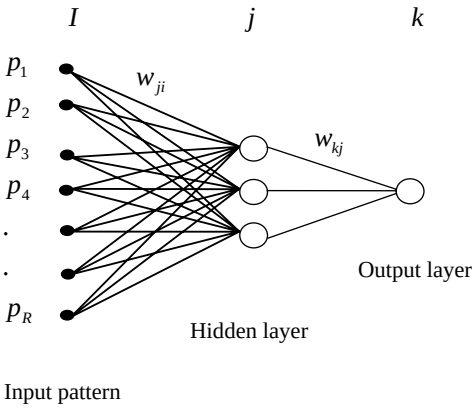

Fig. 1. Structure of a typical feed forward backpropagation neural network.

In terms of topology, an ANN can be classified as a feedforward or feedback (also called recurrent) network [3]. In a feedforward ANN, a unit only sends its output to units from which it does not receive any input directly or indirectly (via other units). In a feedback network, however, feedback is allowed to exist. Feedforward back-propagation is the most widely used neural network paradigm and the one used in this research. In the present work, this type of network has shown to be very efficient in improving the convergence of the genetic algorithm, although the applicability of other types of ANN may also be explored.

Fig. 1 shows the structure of a typical feed forward back-propagation neural network with one hidden layer. Each layer is composed of several neurons and these are fully connected to neurons of the succeeding layer. In each hidden layer and output layer, the processing unit (neuron) sums the input from the previous layer and applies an activation (transfer) function to compute its output to the next layer.

The net input to the jth node in the hidden layer is calculated as [14]:

$$S_j = \sum_{j=1}^{R} w_{ji} p_i \tag{1}$$

where w_{ji} is the connection weight from node i in the input layer to node j in the hidden layer; p_i is the ith input element and R is the number of input features. An activation function is then used to calculate the output of the nodes in the hidden layer. The output of the jth node in the hidden layer is:

$$O_j = f(S_j) \tag{2}$$

where f is the activation function. For a sigmoidal activation function, we have:

$$O_j = \frac{1}{\left(1 + e^{-(S_j + b_j)/\theta_\circ}\right)} \tag{3}$$

where b_j serves as a threshold or bias and θ_0 is the parameter relating to the shape of the sigmoid. Subsequently, the output from the hidden layer is used as input to the output node (or nodes in the next hidden layer). The input and output of the nodes in layer k (Fig. 1) are calculated in the same way:

$$S_k = \sum_{j=1}^{R} w_{kj} O_j \tag{4}$$

$$O_k = f(S_k) \tag{5}$$

On the whole, the output of the network will not be the same as the desired target value.

During cycles of training, the weights of the network are updated in such a way that the difference between the desired response and the computed response is minimized to a required threshold.

In this study, a three-layer back-propagation neural network with hyperbolic tangent sigmoid transfer function and gradient descent learning algorithm with adaptive learning rate was used.

4. Hybrid (intelligent) genetic algorithm

In this paper a hybrid optimization algorithm is presented which is based on the integration of neural network in a conventional genetic algorithm procedure. In the proposed algorithm, the efficiency of genetic algorithm is enhanced by using the learning capabilities of neural network. By combining the two methods, the advantages of both methods are exploited to produce a hybrid optimization method which is both robust and fast. In the algorithm, neural network is used to guide the GA in its search for the optimum solution.

Fig. 2 shows the flowchart of the proposed intelligent genetic algorithm (NeuroGA code). The procedure followed in the proposed algorithm is as follows:

1. The procedure starts with random generation of an initial population.
2. The values of the objective (fitness) function are evaluated for the individuals.
3. The objective function values are then sorted and the best and worst individuals are identified.
4. The neural network is trained using the objective function values and the corresponding design variables.
5. The trained network is used to predict a combination of the variables which could possibly produce an objective function value slightly better than the best objective function value in that generation.
6. The objective function value and the corresponding variables found in step five replace the worst objective function value and the corresponding variables.
7. The processes of selection, crossover and mutation are performed as in a simple genetic algorithm procedure and a new population of individuals (new generation) is created.
8. The new generation replaces the current generation and the entire process is repeated.

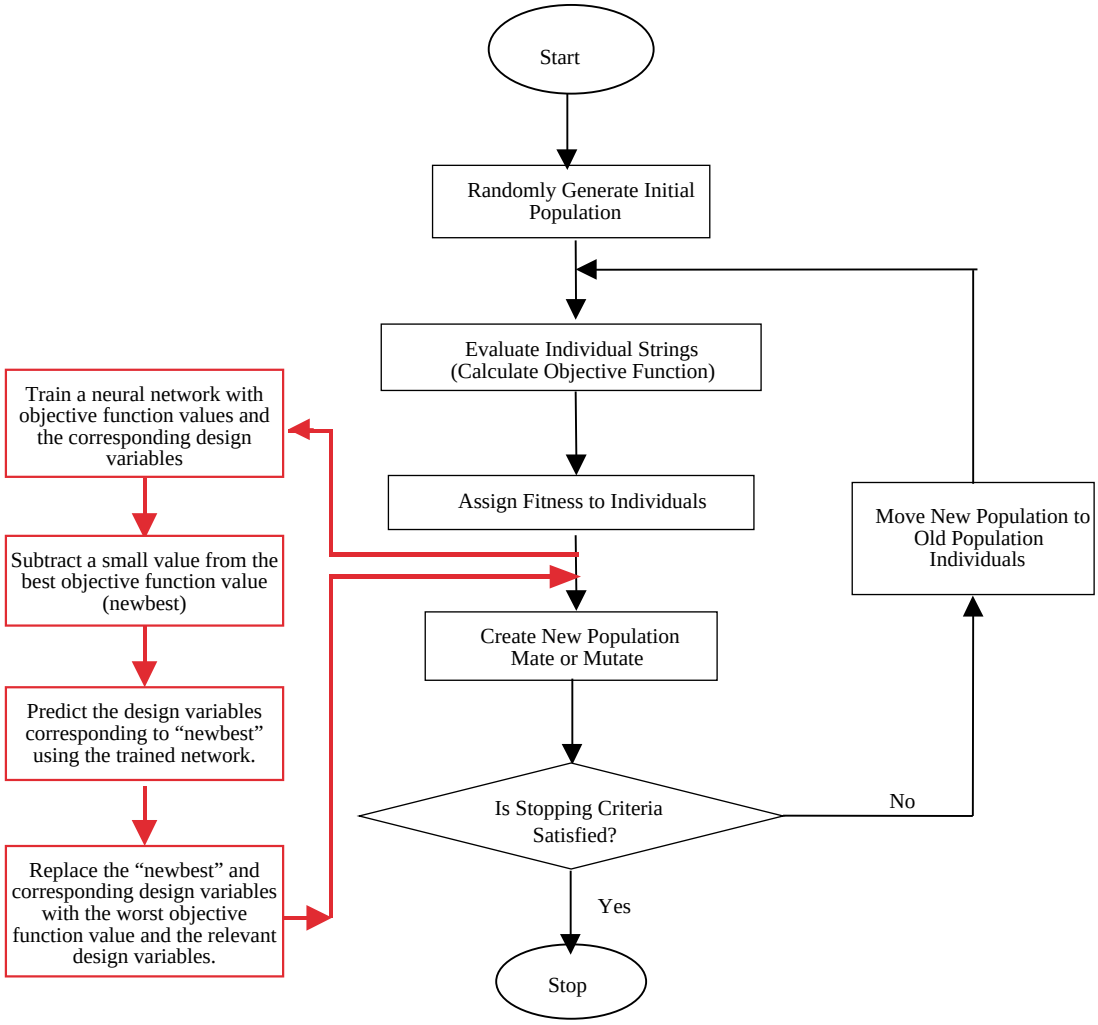

Fig. 2. Flowchart of the proposed hybrid method.

The idea behind the proposed algorithm is that the replacement of a new set of variables which could produce a slightly improved objective function value, with the set of variables producing the worst objective function value could result in improved convergence of the genetic algorithm. In this way, the trained neural network guides the GA in its search to find a better optimum point in each generation. It should be mentioned that, while in the proposed algorithm a better solution is searched and carried to the next generation, the other individuals in the population maintain diversity and prevent premature convergence.

In the following section, the proposed algorithm will be applied to a number of test functions in order to demonstrate that the algorithm can considerably improve the convergence speed of a genetic algorithm as well as the quality of the solution. The results will be compared with those obtained from a conventional genetic algorithm.

5. Numerical examples

To illustrate the computational methodology, the developed intelligent self-learning Genetic Algorithm (NeuroGA program) has been applied to five different commonly used test case problems and the results have been compared with those obtained using a conventional genetic algorithm. The tests were performed with the same GA parameter values (80% probability of crossover, 0.3% probability of mutation and the same population size (10 or 100) for both GA and NeuroGA). For each test case, 10 runs, each starting from a different randomly generated population, have been performed. For consistency of the results, the same random number sequence was used in both algorithms (GA and NeuroGA).

5.1. Rastrigin's function

Rastrigin's function is based on De Jong's first function (continuous, convex and unimodal) with the addition of

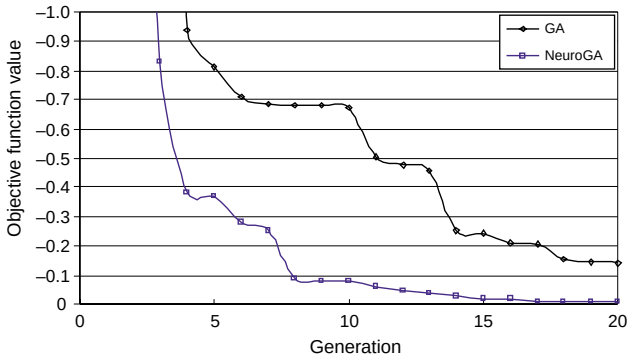

Fig. 3. Comparison of the GA and NeuroGA results averaged over 10 trials (Rastrigin's Problem).

cosine term to produce many local minima. Thus, the test function is highly multimodal. However, the locations of the minima are regularly distributed.

The optimization problem can be formulated as finding a vector **x** that minimizes the objective function:

$$f(x) = 10n + \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i)), \quad n = 2$$

with the following side constraints:

$$-5.12 <= x_i <= 5.12$$

The Multimodal nature of the problem makes it difficult for optimization methods to locate the global optimum and they might get trapped in local optima. The global minimum of $f(x)=0$ is at $x_i=0$.

This problem has been solved using a conventional genetic algorithm and the proposed intelligent genetic algorithm. The results, averaged over 10 trials, are presented in Fig. 3. It is shown that, on average, the conventional GA converged to a minimum value of 0.142 after 20 generations. However, with the proposed hybrid method, this value of objective function was obtained after only seven generations and it was improved significantly in the subsequent generations (0.019 after 15 generations and 0.00091 after 20 generations). It is shown that the convergence rate of NeuroGA is considerably higher than that of GA. The best, worst, mean and standard deviation of the results of 10 different trials of GA and NeuroGA are compared in Table 1.

5.2. Sphere model function

The sphere model function is a widely used multi-modal test function which is formulated as:

$$\text{Minimize} : f(x) = \sum_{i=1}^{n}(x_i - 1)^2$$

with side constraints:

$$-5 \le x_i \le 5$$

The global minimum is $f(x)=0$ which corresponds to $x_i=1.0$, $i=1{:}n$

In the above formulation, n is the number of design variables which defines the dimension of the problem. The problem has been solved with five design variables ($n=5$) using a conventional genetic algorithm as well as the proposed hybrid algorithm. Fig. 4 shows a comparison between the results of the GA and the hybrid method, averaged over 10 trials. It is clearly seen that for the hybrid method the convergence rate was considerably higher and the final objective function value was significantly lower in comparison to the conventional genetic algorithm. On average, GA converged to a minimum objective function value of 1.364 after 50 generations. The hybrid method was able to find a much lower value of 0.015 (nearly the global optimum) after only 43 generations.

5.3. Rosenbrock's function

In order to examine the efficiency of the proposed algorithm in handling constrained optimization problems, the method was applied to the Rosenbrock's valley function with 3 design variables. The Rosenbrock's valley function (De Jong's second function) is a classic optimization problem, also known as the Banana function. The global optimum is inside a long, narrow, parabolic shaped flat valley. To find the valley is trivial, however convergence to the global optimum is difficult. The Rosenbrock's problem can be formulated as finding a vector $\boldsymbol{x}$ that minimizes the objective function:

$$f(x) = \sum_{i=1}^{n-1}\left[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2\right], \quad n = 3$$

Table 1
Results of multiple runs for all test cases

Function	Optimum value	GA				NeuroGA			
		Worst	Best	Average	Standard deviation	Worst	Best	Average	Standard deviation
Rastrigin	0.000	0.939	0.000	0.142	0.287	0.009	0.000	0.001	0.003
Sphere	0.000	3.654	0.539	1.364	0.899	0.086	0.0001	0.015	0.029
Rosenbrock	0.000	119.941	2.366	22.543	36.396	6.359	1.581	8.984	2.112
Constrained	6961.814	5401.490	6866.940	6264.332	474.162	6282.731	6947.118	6635.970	297.734
Schwefel	0.000	190.332	0.525	65.728	66.094	0.785	0.325	0.583	0.192

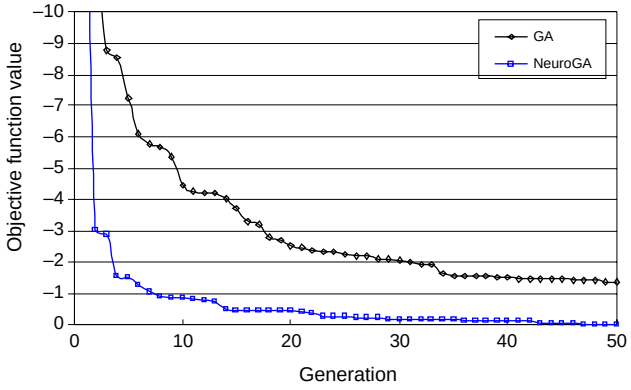

Fig. 4. Comparison of the GA and NeuroGA results averaged over 10 trials (Sphere Function).

with the following constraints:

$$C_1(x) = 1 - \left(\prod_{i=1}^{n} x_i\right)^2 \leq 0 \qquad C_2(x) = \left|n - \left|\sum_{i=1}^{n} x_i\right|\right| = 0$$

and the side constraints:

$$-5.12 \leq x_i \leq 5.12$$

The global minimum is $f(x)=0$ at $x_i=1.0, i=1{:}n$.

The results for the case of Rosenbrock's function with three variables, averaged over 10 trials, are shown in Fig. 5. Comparison of the results indicates that, again, the hybrid method offers a significant improvement in the results compared to the conventional GA. On average, after 50 generations, the minimum objective function value obtained with GA was 22.543 whereas the hybrid method was able to obtain a much lower value of 8.984 after 50 generations. The GA improved the objective function value to 3.675 after 1000 generations. This value was obtained by the NeuroGA only after only 91 generations.

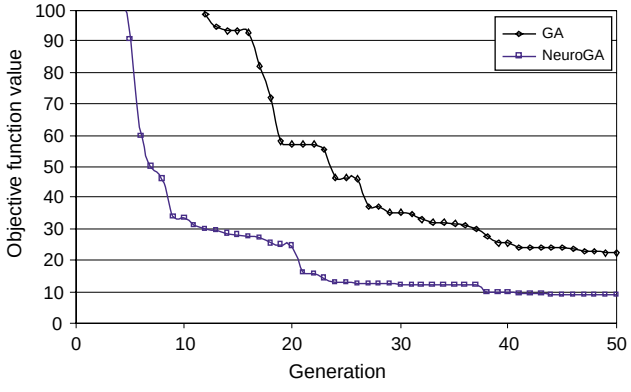

Fig. 5. Comparison of GA and NeuroGA results averaged over 10 trials (Rosenbrock's Function with three variables).

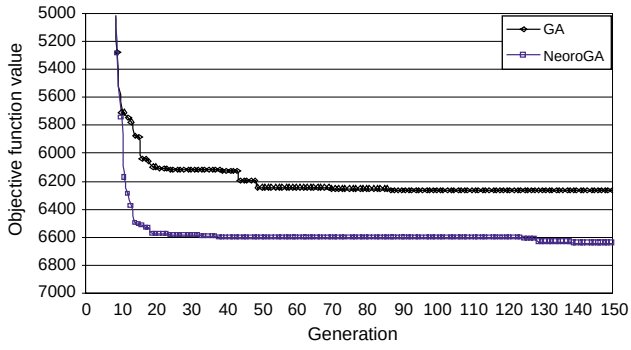

Fig. 6. Comparison of GA and NeuroGA results averaged over 10 trials (Constrained problem [13]).

5.4. Constrained problem

The next example is another constrained minimization problem with two nonlinear inequality constraints [13]. The problem is formulated as:

$$\text{Minimize } f(x) = (x_1 - 10)^3 + (x_2 - 20)^3$$

Subject to nonlinear constraints:

$$(x_1 - 5)^2 + (x_2 - 5)^2 - 100 \geq 0,$$

$$-(x_1 - 6)^2 - (x_2 - 5)^2 + 82.81 \geq 0,$$

and side constraints:

$$13 \leq x_1 \leq 100, \quad 0 \leq x_2 \leq 100.$$

The known global solution is $\boldsymbol{x}=(14.095, 0.84296)$ and the corresponding objective function value $f(x)=-6961.81381$. Both constraints are active at the optimum [13].

Fig. 6 shows a comparison between the results of the GA and the hybrid method. Again it is clearly seen that for the hybrid method the convergence rate was considerably higher and the final objective function value was significantly lower in comparison to the conventional genetic algorithm.

5.5. Schwefel's function

This is a multi-modal test function, mathematically formulated as:

$$\text{Minimize } 418.9829n + \sum_{i=1}^{n}\left(-x_i \sin\left(\sqrt{|x_i|}\right)\right), \quad n = 10$$

with side constraints:

$$-500 \leq x_i \leq 500$$

The global minimum is $f(x)=0$ at $x_i=420.9687$ ($i=1{:}n$). The problem was solved with 10 design variables (10 dimensional space) and the results, averaged over

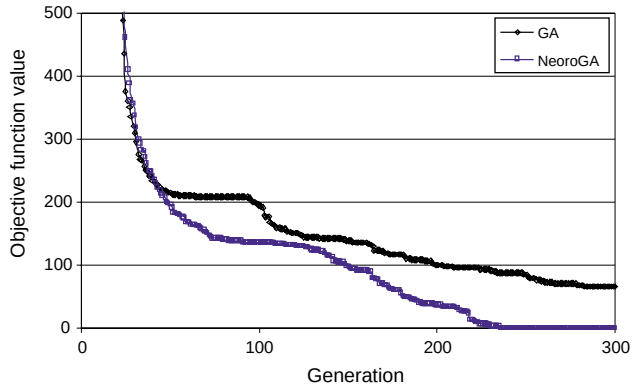

Fig. 7. Comparison of the GA and NeuroGA results averaged over 10 trials (Schwefel Function).

10 trials, are shown in Fig. 7. On average, the best (minimum) objective function value obtained using the GA after 300 generations was 65.728 whereas the hybrid method was able to obtain much improved solutions (very close to the global optimum) of 0.696 after 247 generations and 0.583 after 300 generations.

The above examples, show the advantages of the proposed hybrid algorithm over the conventional genetic algorithm in terms of significant improvements in both the convergence rate and solution quality. The improvements become more significant in the case of more complicated optimization problems.

6. General performance of the algorithm:

The results indicate that the algorithm described here found much better solutions in all trials and for all tests cases, in a smaller number of generations compared with conventional genetic algorithm. The best and worst solutions together with the mean and standard deviation of results of the trial solutions for the test functions are shown in Table 1. A comparison of the results for the two algorithms indicates that the self-learning element of NeuroGA has resulted in considerable improvements in the best and average solutions and a significant reduction in the number of generations required to obtain the best solution. It should be noted that the standard deviations for the tests results obtained using the NeuroGA algorithm are small (Table 1). This is an important characteristic for the application of the algorithm to the solution of 'real world' problems where cost and time constraints prohibit repeated runs of the algorithm [7]. The small standard deviations for the algorithm described here indicate that it is robust in finding a near optimum solution without the need for repeated runs of the optimization. This is very important when the objective function is evaluated following computationally expensive system simulations.

7. Conclusions

A hybrid intelligent genetic algorithm is presented which is based on the integration of neural network and genetic algorithm. In the proposed algorithm, the efficiency of genetic algorithm is enhanced by using the learning capabilities of neural network. By combining the two methods, the advantages of both methods are exploited to produce a hybrid optimization method which is both robust and fast. In the algorithm, a back-propagation neural network is used to improve the convergence of the genetic algorithm in search for global optimum. The efficiency of the approach has been illustrated by application to a number of test cases involving nonlinear, multimodal and constrained optimization problems. For each test case, multiple runs, each starting from a different randomly generated population, have been performed. The results show that integration of the neural network in the genetic algorithm procedure can yield significant improvements in both the convergence rate and solution quality.

Application of the presented algorithm to some real world problems is the subject of current research by the authors. The authors are also currently working on some applications of the developed method to problems with high dimensionality and it is envisaged that some modifications might be necessary when the dimensionality of the problems becomes very high.

In the present work, feed forward back propagation neural network has shown to be very efficient in improving the convergence of the genetic algorithm, although the applicability of other types of ANN may also be explored.

References

[1] Abu-Alola AH, Gough NE. Identification and adaptive control of nonlinear processes using combined neural network and genetic algorithms. In: Pearson DW, Steele NC, Albrecht RF, editors. Proceedings of the international conference on artificial neural nets and genetic algorithms. Verlag: Springer; 1995. p. 396–9.

[2] Adeli H, Hung SL. Machine learning; neural networks, genetic algorithms and fuzzy systems. New York: Wiley; 1995.

[3] Bishop C. Neural network for pattern recognition. Oxford: Clarendon Press; 1995.

[4] Davis L. Handbook of genetic algorithms. New York: Van Nostrand Reinhold; 1991.

[5] Deb K, Goel T. Controlled elitism non-dominated sorting genetic algorithms for better convergence First international conference on evolutionary multi-criterion optimization 2000 p. 67–81.

[6] Deb K, Agrawal S, Pratap A, Meyaruvan T. (2000), A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II, Lecture Notes in Computer Science, Iss. 1917, 849–858.

[7] Farmani R, Wright JA. Self-adaptive fitness formulation for constrained optimization. IEEE Trans Evol Comput 2003;7(5): 445–55.

[8] Goldberg DE. Genetic algorithms in search, optimization, and machine learning. USA: Addison-Wesley; 1989.

[9] Holand JH. Adaptation in natural and artificial systems, ANN Arbor. Michigan: The University of Michigan Press; 1975.
[10] Javadi AA. Estimation of air losses from tunnels driven under compressed air using neural networks. Proceedings of the 10th international conference on computer methods and advances in geomechanics, Tuscon, Arizona. vol.1 2001 p. 207–211.
[11] Javadi AA, Brown MJ, Hyde AFL. Prediction of load bearing capacity of piles from rapid load pile testing data using neural networks. Proceedings of the 5th international conference on deep foundation practice, Singapore 2001 p. 235–242.
[12] Javadi AA, Farmani R, Tan TP. An intelligent self-learning genetic algorithm Proceedings of the 11th international EG-ICE workshop, Weimar 2001 p. 26–35.
[13] Koziel S, Michalewicz Z. Evolutionary algorithms, homomorphous mapping, and constrained parameter optimization. J Evol Comput 1999;7(1):19–44.
[14] Pao YH. Adaptive pattern recognition and neural networks. USA: Addison-Wesley Publishing Company; 1989.
[15] Wasserman PD. Neural computing: theory and practice. New York: Van Nostrand Reinhold; 1988.