Proceedings of International Joint Conference on Neural Networks, Montreal, Canada, July 31 - August 4, 2005

Real-Coded Genetic Algorithm with Average-Bound Crossover and Wavelet Mutation for Network Parameters Learning

S.H. Ling
Centre of Multimedia Signal Processing,
Department of Electronic and Information Engineering,
The Hong Kong Polytechnic University, Hung Hom,
Kowloon, Hong Kong

F.H.F. Leung
Centre of Multimedia Signal Processing,
Department of Electronic and Information Engineering,
The Hong Kong Polytechnic University, Hung Hom,
Kowloon, Hong Kong

Abstract – This paper presents the learning of neural network parameters using a real-coded genetic algorithm (RCGA) with proposed crossover and mutation. They are called the average-bound crossover (AveBXover) and wavelet mutation (WM). By introducing the proposed genetic operations, both the solution quality and stability are better than the RCGA with conventional genetic operations. A suite of benchmark test functions are used to evaluate the performance of the proposed algorithm. An application example on an associative memory neural network is used to show the learning performance brought by the proposed RCGA.

I. Introduction

Learning or training is one of the important issues of neural networks. The learning process aims to find a set of optimal network parameters. The widely-used gradient methods [1-2], such as MRI, MRII, MRIII rules, and back-propagation techniques, adjust the network parameters based on the gradient information of the fitness function in order to reduce the mean square error over all input patterns. One major weakness of the gradient methods is that the derivative information of the fitness function is needed, meaning it has to be continuous. The learning process is easily trapped in a local optimum, especially when the problems are multimodal and the learning rules are network structure dependent. To tackle this problem, the real-code genetic algorithm (RCGA) [4] was proposed for optimization problems in a large, complex, non-differentiable and multimodal domain [11]. RCGA is a good training algorithm for neural or neural-fuzzy networks [5-6]. The same RCGA can be used to train many different networks regardless of whether they are feed-forward, recurrent, or of other structure types. This generally saves a lot of human efforts in developing training algorithms for different types of networks.

A lot of research efforts have been spent to improve the performance of RCGA. Basically, RCGA involves two genetic operations: crossover and mutation. Recently, different crossover operations for RCGA have been proposed to improve the efficiency of the algorithm. Unimodal normal distribution crossover (UNDX) was proposed by Ono *et. al.* [7] for handling multimodal functions and non-separability problems. UNDX mixes the parental information and shows a good search ability. However, it changes the fundamental concept that the crossover operation combines the parents to generate offspring, not mixing the parents. Blend crossover (BLX-α) was proposed by Eshelman *et. al.* [8], which shows a good search ability for separable functions. It combines the parents to generate offspring. However, BLX-α has difficulty in handling non-separability optimisation problems. Also, the above crossover operations are not suitable to handle optimisation problems with the optimal point located near the domain boundary. For mutation operations, uniform mutation and non-uniform mutation can be found [4]. Uniform mutation is to change the value of a randomly selected gene to a value between its upper and lower bounds. Non-uniform mutation is capable of fine-tuning the parameters by increasing or decreasing the value of a randomly selected gene by a weighted random number. The weight is usually a monotonic decreasing function of the number of iteration.

In this paper, new genetic operations of crossover and mutation are proposed. The crossover operation is called the average-bound crossover (AveBXover), which combines the average crossover and bound crossover. The average crossover manipulates with the selected parents, the minimum and maximum values of the genes. The bound crossover is capable of moving the offspring near the domain boundary. On realizing the AveBXover operation, the offspring spreads over the domain so that a higher chance of reaching the global optimum can be obtained. The proposed mutation operation is called the wavelet mutation (WM), which applies the wavelet theory [9]. Wavelet is a tool to model seismic signals by combining dilations and translations of a simple, oscillatory function (mother wavelet) of finite duration. The wavelet function has two properties: 1) the function integrates to zero, and 2) it is square integrable, or equivalently has finite energy. Owing to the properties of the wavelet, the convergence and solution stability are improved. By introducing these genetic operations, the GA performs more efficiently and provides a faster convergence than the GA with conventional genetic operations in a suite of six benchmark test functions [10]. In addition, the GA with the proposed operations gives smaller standard deviation of results, i.e. the solution quality of the GA with the proposed operations is more stable.

This paper is organized as follows. Section II presents the RCGA with the proposed genetic operations. Six benchmark test functions are used to evaluate the performance of the proposed method in section III. An application example on associative memory is given in section IV. A conclusion will be drawn in section V.

0-7803-9048-2/05/$20.00 ©2005 IEEE

II. AVERAGE-BOUND CROSSOVER AND WAVELET MUTATION FOR RCGA

The Real-Coded Genetic Algorithm (RCGA) process [3-4] is shown in Fig. 1. First, a set of population of chromosomes P is created. Each chromosome $\mathbf{p}$ contains some genes (variables). Second, the chromosomes are evaluated by a defined fitness function. The better chromosomes will return higher values in this process. Third, some of the chromosomes are selected to undergo genetic operations for reproduction by the method of normalized geometric ranking [3]. Normalized geometric ranking is a ranking selection function based on the non-stationary penalty function. The non-stationary penalty is a function of the generation number; as the number of generations increases so does the penalty. Therefore, as the penalty increases it puts more and more selective pressure on the RCGA to find a feasible solution. Fourth, genetic operations of crossover are performed. The crossover operation is mainly for exchanging information between two parents that are obtained by a selection operation. In the crossover operation, one of the parameters is the probability of crossover p_c which gives us the expected number $p_c \times pop\_size$ (where pop_size is the number of chromosomes in the population) of chromosomes that undergo the crossover operation.

We propose a new crossover. First, four chromosomes will be generated (instead of two chromosomes in the conventional GA). Second, the best two offspring in terms of the fitness value will be selected to replace their parents. After the crossover operation, the mutation operation follows. It operates with the parameter of the probability of mutation (p_m). The mutation operation is to change the genes of the chromosomes in the population such that the features inherited from their parents can be changed. After going through the mutation operation, the new offspring will be evaluated using the fitness function. The new population will be formed when the new offspring replaces the chromosome with the smallest fitness value. After the operations of selection, crossover and mutation, a new population is generated. This new population will repeat the same process. Such an iterative process will be terminated when a defined condition is met. The details about the proposed crossover and mutation operations are given as follows.

A. Average-bound crossover

The crossover operation is mainly for exchanging information from the two parents, chromosomes $\mathbf{p}_1$ and $\mathbf{p}_2$, obtained in the selection process. The two parents will finally produce two offspring. The average-bound crossover (AveBXover) comprises two operations: average crossover and bound crossover. The details of the crossover operation are as follows,

Average crossover

$$\mathbf{o}_{s_c^1} = (\mathbf{p}_1 + \mathbf{p}_2)/2 \tag{1}$$

$$\mathbf{o}_{s_c^2} = ((\mathbf{p}_{\max} + \mathbf{p}_{\min})(1 - w_a) + (\mathbf{p}_1 + \mathbf{p}_2)w_a)/2 \tag{2}$$

Bound crossover

$$\mathbf{o}_{s_c^3} = \mathbf{p}_{\max}(1 - w_b) + \max(\mathbf{p}_1, \mathbf{p}_2)w_b \tag{3}$$

$$\mathbf{o}_{s_c^4} = \mathbf{p}_{\min}(1 - w_b) + \min(\mathbf{p}_1, \mathbf{p}_2)w_b \tag{4}$$

```
Procedure of the RCGA
begin
    τ←0  // τ : iteration number
    Initialize P(τ)      //P(τ) : population for iteration τ
    Evaluate f(P(τ))     // f(P(τ)) :fitness function
while (not termination condition) do
    begin
        τ←τ+1
        Perform selection operation
        Determine the number of crossover based on p_c
        Select 2 parents p_1 and p_2 from P(τ-1)
        Perform crossover operation
        Four chromosomes will be generated
        Select the best 2 offspring in terms of the fitness value
        Perform mutation operation for the whole population
        based on p_m
        Reproduce a new P(τ)
        Evaluate f(P(τ))
    end
end
```

Fig.1. RCGA process

where

$$\mathbf{o}_{s_c^k} = \left[o_{s_1^k} \quad o_{s_2^k} \quad \cdots \quad o_{s_{no\_vars}^k} \right], k = 1, 2, 3, 4.$$

$$\mathbf{p}_i = \left[p_{i_1} \quad p_{i_2} \quad \cdots \quad p_{i_j} \quad \cdots \quad p_{i_{\text{no\_vars}}} \right], \; i = 1, 2; \; j = 1, 2, \ldots, no\_vars, \tag{5}$$

$$para^j_{\min} \le p_{i_j} \le para^j_{\max}, \tag{6}$$

$$\mathbf{p}_{\max} = \left[para^1_{\max} \quad para^2_{\max} \quad \cdots \quad para^{no\_vars}_{\max} \right], \tag{7}$$

$$\mathbf{p}_{\min} = \left[para^1_{\min} \quad para^2_{\min} \quad \cdots \quad para^{no\_vars}_{\min} \right], \tag{8}$$

where no_vars denotes the number of variables to be tuned; $para^j_{\min}$ and $para^j_{\min}$ are the minimum and maximum values of p_{i_j} respectively for all i; $w_a, w_b \in [0 \;\; 1]$ denotes the weight of average crossover and bound crossover to be determined by users respectively, $\max(\mathbf{p}_1, \mathbf{p}_2)$ denotes the vector with each element obtained by taking the maximum between the corresponding element of $\mathbf{p}_1$ and $\mathbf{p}_2$. For instance, $\max([1 \;\; -2 \;\; 3], [2 \;\; 3 \;\; 1]) = [2 \;\; 3 \;\; 3]$. Similarly, $\min(\mathbf{p}_1, \mathbf{p}_2)$ gives a vector by taking the minimum value. For instance, $\min([1 \;\; -2 \;\; 3], [2 \;\; 3 \;\; 1]) = [1 \;\; -2 \;\; 1]$. Among $\mathbf{o}_{s_c^1}$ to $\mathbf{o}_{s_c^4}$, the two with the largest fitness value are used as the offspring of the crossover operation. These two offspring are put back into the population to replace their parents.

The rationale behind the AveBXover is that if the offspring spreads over the domain, a higher chance of reaching the global optimum can be obtained. As seen from (1) to (4): (1) and (2) will move the offspring near the centre region of the concerned domain (as w_a in (2) approaches 1, $\mathbf{o}_{s_c^2}$ approaches $(\mathbf{p}_1 + \mathbf{p}_2)/2$, which is the average of the selected parents; and as w_a approaches 0, $\mathbf{o}_{s_c^2}$ approaches $(\mathbf{p}_{\max} + \mathbf{p}_{\min})/2$, which is the average of

the domain boundary), while (3) and (4) will move the offspring near the domain boundary (as w_b in (3) and (4) approaches 0, $\mathbf{o}_{s_c^3}$ and $\mathbf{o}_{s_c^4}$ approaches $\mathbf{p}_{\max}$ and $\mathbf{p}_{\min}$ respectively). The result of the crossover depends on the

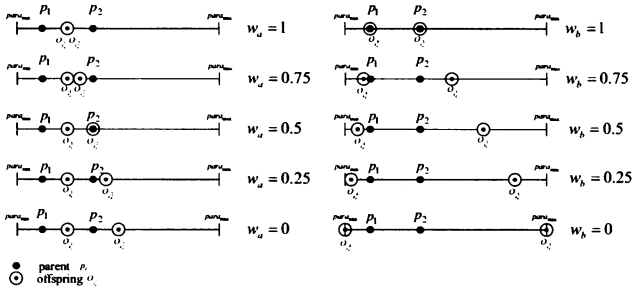

Fig.2. Parents and offspring under different values of the weights w_a and w_b (w_a , w_b = 0, 0.25, 0.5, 0.75 and 1.)

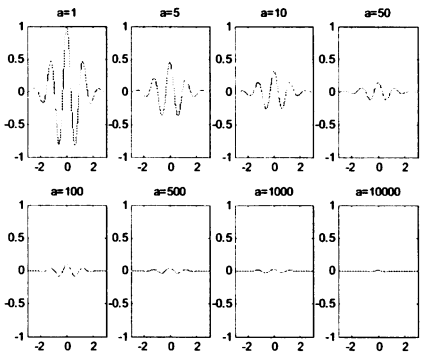

Fig. 3. A Morlet wavelet dilated by various values of the parameter a (x-axis: x, y-axis: $\psi_{a,0}(x)$.)

values of the weights w_a and w_b. They vary with the optimisation problem and are chosen by trial and error. Fig. 2 shows the relationship between the parents and the offspring under different values of the weights. In this figure, the line represents the domain of a gene. The end points of the line represent the minimum and maximum values of the gene. The dot (•) represents the parents and the circle-dot (⊙) represents the offspring. According to (1) to (4), the offspring $o_{s_c^i}$ are generated. We can see how the offspring spreads over the domain under different values of w_a and w_b.

B. Wavelet Mutation

Before presenting the wavelet mutation operation, we first discuss the wavelet theory.

B.1 Wavelet theory

Certain seismic signals can be modelled by combining translations and dilations of an oscillatory function with finite duration called a "wavelet". A continuous time function $\psi(x)$ is called a "mother wavelet" or "wavelet" if it satisfies the following properties:

Property 1:

$$\int_{-\infty}^{+\infty}\psi(x)dx = 0 \tag{9}$$

In other words, the total positive energy of $\psi(x)$ is equal to the total negative energy of $\psi(x)$.

Property 2:

$$\int_{-\infty}^{+\infty}|\psi(x)|^2 dx < \infty \tag{10}$$

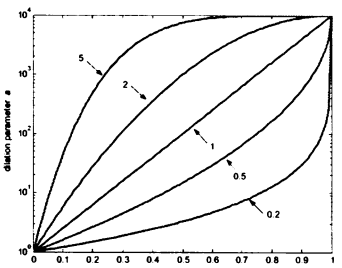

Fig. 4. The effect of the shape parameter ζ to a.

where most of the energy in $\psi(x)$ is confined to a finite duration and bounded. Morlet wavelet is an example mother wavelet, which is proposed by Daubechies [9]:

$$\psi(x) = e^{-x^2/2}\cos(5x) \tag{11}$$

The Morlet wavelet integrates to zero (Property 1). Over 99% of the total energy of the function is contained in the interval of $-2.5 \le x \le 2.5$ (Property 2).

In order to control the magnitude and the position of $\psi(x)$, $\psi_{a,b}(x)$ is defined as:

$$\psi_{a,b}(x) = \frac{1}{\sqrt{a}}\psi\left(\frac{x-b}{a}\right) \tag{12}$$

where a is the dilation parameter and b is the translation parameter. Notice that

$$\psi_{1,0}(x) = \psi(x) \tag{13}$$

As

$$\psi_{a,0}(x) = \frac{1}{\sqrt{a}}\psi\left(\frac{x}{a}\right), \tag{14}$$

it follows that $\psi_{a,0}(x)$ is an amplitude-scaled version of $\psi(x)$. Fig. 3 shows different dilations of the Morlet wavelet. The amplitude of $\psi_{a,0}(x)$ will be scaled down as the dilation parameter a increases.

B.2 Wavelet Mutation

We propose a mutation operation based on the wavelet theory, and is called the "Wavelet Mutation" (WM). The details of the operation are as follows. Every gene of the chromosomes will have a chance to mutate governed by a probability of mutation, $p_m \in [0\ \ 1]$, which is defined by the user. This probability gives an expected number ($p_m \times pop\_size \times no\_vars$) of genes that undergo the mutation. For each gene, a random number between 0 and 1 will be generated such that if it is less than or equal to p_m, the mutation will take place on that gene which is updated instantly. If $\mathbf{o}_s = [o_{s_1},\ o_{s_2},\ \ldots\ , o_{s_{no\_vars}}]$ is the selected chromosome and the element o_{s_j} is randomly

selected for mutation (the value of o_{s_j} is inside $[\,para^j_{\min}, para^j_{\max}\,]$), the resulting chromosome is given by $\hat{\mathbf{o}}_{\mathbf{s}} = \left[o_{s_1}, \ \ldots \ , \hat{o}_{s_j}, \ \ldots \ , o_{s_{no\_vars}}\right]$, where $j \in 1, 2, \ldots no\_vars$, and

$$\hat{o}_{s_j} = \begin{cases} o_{s_j} + \delta \times \left(para^j_{\max} - o_{s_j}\right) \text{ if } \delta > 0 \\ o_{s_j} + \delta \times \left(o_{s_j} - para^j_{\min}\right) \text{ if } \delta \le 0 \end{cases}, \tag{15}$$

$$\delta = \psi_{a,0}(\varphi) \tag{16}$$

$$\delta = \frac{1}{\sqrt{a}} \psi\left(\frac{\varphi}{a}\right) \tag{17}$$

By using the Morlet wavelet in (11) as the mother wavelet,

$$\delta = \frac{1}{\sqrt{a}} e^{-\left(\frac{\varphi}{a}\right)^2 / 2} \cos\left(5\left(\frac{\varphi}{a}\right)\right) \tag{18}$$

If δ is positive ($\delta > 0$) approaching 1, the mutated gene will tend to the maximum value of o_{s_j}. Conversely, when δ is negative ($\delta \le 0$) approaching −1, the mutated gene will tend to the minimum value of o_{s_j}. A larger value of $|\delta|$ gives a larger searching space for o_{s_j}. When $|\delta|$ is small, it gives a smaller searching space for fine-tuning the gene. Referring to Property 1 of the wavelet, the total positive energy of the mother wavelet is equal to the total negative energy of the mother wavelet. Then, the sum of the positive δ should be equal to the sum of the negative δ when the number of samples is large. That is,

$$\frac{1}{N}\sum_{N} \delta = 0 \ \text{ for } N \to \infty, \tag{19}$$

where N is the number of samples.

Hence, the overall positive mutation and the overall negative mutation throughout the evolution are nearly the same. As over 99% of the total energy of the mother wavelet function is contained in the interval [−2.5, 2.5], φ can be generated from [−2.5, 2.5] randomly. The value of the dilation parameter a can be set to vary with the value of τ/T in order to meet the fine-tuning purpose, where T is the total number of iteration and τ is the current number of iteration. In order to perform a local search when τ is large, the value of a should increase as τ/T increases so as to reduce the significance of the mutation. Hence, a monotonic increasing function governing a and τ/T is proposed as follows.

$$a = e^{-\ln(g) \times \left(1 - \frac{\tau}{T}\right)^{\zeta} + \ln(g)} \tag{20}$$

where ζ is the shape parameter of the monotonic increasing function, g is the upper limit of the parameter a. In this paper, g is set as 10000. The effects of the various values of the shape parameter ζ to a with respect to τ/T are shown in Fig. 4. The value of a is between 1 and 10000. Referring to (18), the maximum value of δ is 1 when the random number of φ =0 (and a = 1). Then referring to (15), the offspring gene $\hat{o}_{s_j} = o_{s_j} + 1 \times \left(para^j_{\max} - o_{s_j}\right)$. It ensures that a large search space for the mutated gene is given. When the value τ/T is near to 1, the value of a is so large that the maximum value of δ will become very small. For example, at $\tau/T = 0.9$ and $\zeta = 1$, the dilation parameter a = 4000. If the random value of φ is zero, the value of δ will be equal to 0.0158. With $\hat{o}_{s_j} = o_{s_j} + 0.0158 \times \left(para^j_{\max} - o_{s_j}\right)$, a small searching space for the mutated gene is given for fine-tuning.

III. Experimental Studies and Analysis

A. Benchmark test functions and experiment setup

A suite of six benchmark test functions [10] are used to test the performance of the GA with the proposed genetic operations. Many different kinds of optimization problems are covered by these benchmark test functions. They are divided into three categories: unimodal functions, multimodal functions with only a few local minima, and multimodal functions with many local minima. The six benchmark test functions are detailed in Table I. They can test the searching ability of the proposed algorithm comprehensively. In other words, the proposed algorithm is not biased to some chosen problem. To avoid the crossover operation having a strong bias to that $(\mathbf{p}_{\max} + \mathbf{p}_{\min})/2$ is also the location of the optimum, in this paper, the ranges of the domain boundary of some test functions are different from those in [10] in order to shift the location of the optima.

The crossover operation for comparison is the UNDXBXover, which consists of 2 published crossover operations: Unimodal normal distribution crossover (UNDX) [7] and Blend crossover (BLX-α) [8]. The mutation operation for comparison is the non-uniform mutation (NUM) [4]. The population size is set at 100. All the results are the averaged ones out of 50 runs. The weight w_a of the AveBXover is set at 0.5 for all functions. The weight w_b is set at 0.5 for f_1 to f_4 and f_6, w_b=1 for f_5. The parameter ζ for WM is chosen by trial and error through experiments for good performance. ζ is set at 5, 5, 0.5, 0.5, 2, and 5 for f_1 to f_6 respectively. The probability of crossover is set at 0.8 and the probability of mutation is set at 0.5 for f_1 to f_3 and f_6, 0.8 for f_4 and f_5. In this paper, RCGA with Avergae-Bound Crossover and Wavelet Mutation (AveBXover+WM), RCGA with Avergae-Bound Crossover and Non-Uniform Mutation (AveBXover+NUM), RCGA with Unimodal Normal Distribution and Blend Crossover and Wavelet Mutation, (UNDXBXover+WM), and RCGA with Unimodal Normal Distribution and Blend crossover and Non-Uniform Mutation (UNDXBXover+NUM) are used to test the benchmark test functions.

TABLE I. BENCHMARK TEST FUNCTIONS

Test function	Domain range	Optimal point
$f_1(\mathbf{x}) = \sum_{i=1}^{30} x_i^2$	$-50 \le x_i \le 150$	$f_1(\mathbf{0}) = 0$
$f_2(\mathbf{x}) = \sum_{i=1}^{29}\left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right]$	$-2.048 \le x_i \le 2.048$	$f_2(\mathbf{1}) = 0$
$f_3(\mathbf{x}) = \sum_{i=1}^{30}(\lfloor x_i + 0.5 \rfloor)^2$	$-5 \le x_i \le 10$	$f_3(\mathbf{0}) = 0$
$f_4(\mathbf{x}) = -\cos(x_1)\cdot\cos(x_2)$ $\cdot\exp\left(-\left((x_1-\pi)^2 + (x_2-\pi)^2\right)\right)$	$-300 \le x_1, x_2 \le 300$	$f_4([\pi,\ \pi])$ $= -1$
$f_5(\mathbf{x}) = \sum_{i=1}^{9}\left[a_i - \frac{x_1\left(b_i^2 + b_i x_2\right)}{b_i^2 + b_i x_3 + x_4}\right]^2$	$-5 \le x_i \le 5$	$f_5(0.1928,\ 0.1908,$ $0.1231,\ 0.1358)$ ≈ 0.0003075
$f_6(\mathbf{x}) = \frac{1}{4000}\sum_{i=1}^{30} x_i^2 - \prod_{i=1}^{30}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$-1200 \le x_i \le 600$	$f_6(\mathbf{0}) = 0$

B. Experiment results

The experiment results in terms of the mean fitness value, best fitness value, standard deviation, and the *t*-test value for f_1 to f_6 are tabulated in Table. II. The comparison between different genetic operations on f_1 to f_6 is shown in Fig. 5. The *t*-test is a statistical method to evaluate the significant difference between two algorithms. The *t*-value will be negative if the first algorithm is better than the second. When the *t*-value is more than −1.645 (degree of freedom = 49), there is a significant difference between the two algorithms with a 95% confidence level. f_1 to f_3 are unimodal functions. f_1 is a sphere model which is probably the most widely used test function. It is smooth and symmetric. The performance on this function is a measure of the convergence rate of a searching algorithm. For f_1, the results in terms of the mean and the best fitness value of AveBXover with WM or NUM are better than those of the corresponding UNDXBXover. Comparing AveBXover with WM to UNDXBXover with WM, the mean fitness 4-time is better. A much smaller standard deviation is given by the AveBXover, which means the solution is more stable. Comparing the mutation operations WM and NUM, the proposed WM is more effective than NUM in term of the fitness value and standard deviation. Both the solution quality and stability of WM are better than those of NUM. In addition, the *t* value is −10.62, which implies that the proposed genetic operations (AveXover with WM) are better than the conventional genetic operations (UNDXBXover with NUM). In Fig. 5, AveBXover with WM displays a faster convergence rate than UNDXBXover with NUM thanks to its better searching ability. f_2 is strongly non-separable and the optimum is located in a very narrow ridge. The tip of the ridge is very sharp, and it runs around a parabola. Algorithms that are unable to discover good searching directions will perform poorly in this problem. The proposed algorithm (AveBXover with WM) outperforms the UNDXBXover with NUM. The *t* value is −313.3. Although the best fitness values on using WM with different crossover operations are a bit worse than those on using NUM, the mean value, standard deviation and convergence rate offered by WM are better. f_3 is a step function that is a representative of flat surfaces. UNDXBXover performs poorly for f_3 because it mainly searches in a small local neighbourhood, but the flat surfaces do not give any searching direction for UNDXBXover. On the other hand, the proposed AveBXover is good for f_3 because it can generate longer jump than UNDXBXover. Comparing WM to NUM with UNDXBXover, the former also gives a better solution. f_4 to f_5 are multimodal functions with only a few local minima. In these 2 functions, we find statistically different results from the proposed genetic operations and the conventional genetic operations. The proposed AveBXover performs better than the conventional one. In addition, the results offered by WM are better than those of NUM in terms of the mean and the best fitness values. Furthermore, WM gives a faster convergence rate. f_6 is a multimodal function with many local minima. It can be seen from Table II that the mean results and the best results offered by the proposed genetic operations are better than those offered by the conventional genetic operations. Also, the solutions have smaller standard deviations. Therefore, in terms of the solution quality and stability, the proposed genetic operations are better than the conventional operations. From Fig. 5, we can see that the convergence rate of the proposed genetic operations is better than that of the conventional genetic operations.

IV. APPLICATION EXAMPLE

Application example on tuning associative memory is given in this section. The associative memory, which maps its input vector into itself, has ten inputs and ten outputs. Thus, the desired output vector is its input vector. 50 input vectors are used for the learning. The associative memory is given by:

$$y_k(t) = \sum_{j=1}^{10} w_{jk} z_j(t),\ k = 1, 2, \ldots, 10 \tag{21}$$

where $z(t)$ is the input vector and w_{jk} is the weight of the link between the input and the output. The objective is to minimize the mean square error (MSE), which is defined as follows:

$$\text{MSE} = \frac{\sum_{k=1}^{10}\sum_{t=1}^{50}\left(z_k(t) - y_k(t)\right)^2}{10 \times 50} \tag{22}$$

The initial range of the weight w_{jk} is between -2 to 2. The experimental results are tabulated in Table III, and the comparison between different genetic operations is shown in Fig. 6. As can be seen from the table, the mean and the best fitness value offered by AveBXover and WM are better. In Addition, the smaller standard deviation implies a more stable solution. The *t*-value for this function is −24.67, which is a relatively large figure. In short, the improved GA is good for training associative memory.

V. CONCLUSION

RCGA with improved genetic operations (crossover and mutation) has been presented. A suit of benchmark test functions has been used to illustrate the merits of the improved genetic operations. An example on associative memory has also been given.

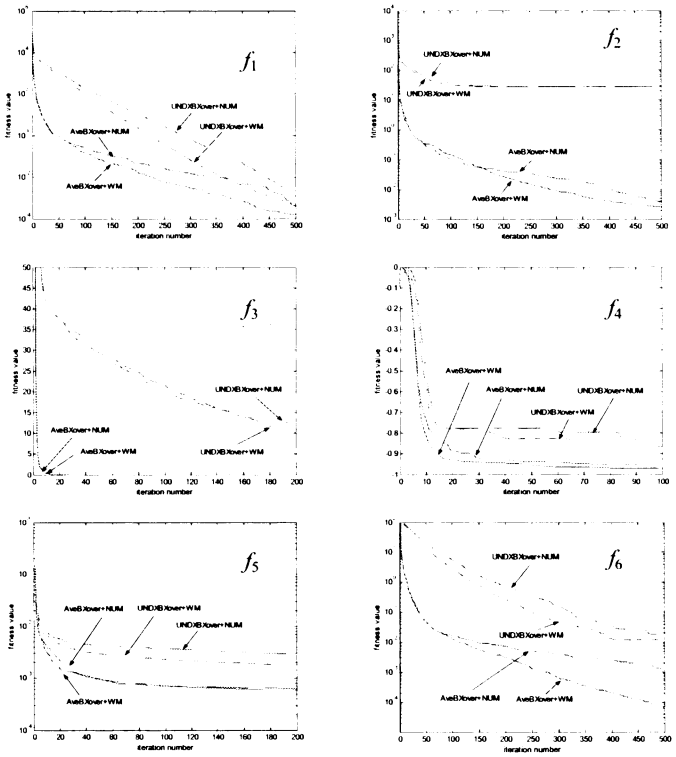

Fig.5. Comparisons between different genetic operations for f_1 to f_6.

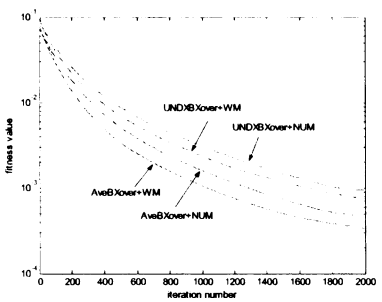

Fig.6. Comparisons between different genetic operations for tuning an associative memory.

ACKNOWLEDGMENT

The work described in this paper was substantially supported by a grant from the Hong Kong Polytechnic University (PhD student Account Code RG9T).

REFERENCES

[1] B. Widrow and M.A. Lehr, "30 years of adaptive neural networks: Perceptron, madaline, and backpropagation," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1415-1442, Sept. 1990.

[2] J.M. Zurada, *Introduction to Artificial Neural Systems*. West Info Access, 1992.

[3] J. Joines and C. Houck, "On the use of non-stationary penalty functions to solve constrained optimizationproblems with genetic algorithm," in *Proc. 1994 Int. Symp. Evolutionary Computation*, Ordando, 1994, pp. 579-584.

[4] Z. Michalewicz, *Genetic Algorithm + Data Structures = Evolution Programs*, 2nd extended ed. Springer-Verlag, 1994.

[5] F.H.F. Leung, H.K. Lam, S.H. Ling, and P.K.S. Tam, "Tuning of the structure and parameters of neural network using an improved genetic algorithm," *IEEE Trans. Neural Networks*, vol.14, no. 1, pp.79-88, Jan. 2003.

[6] S.H. Ling, F.H.F Leung, H.K. Lam, and P.K.S. Tam, "Short-term electric load forecasting based on a neural fuzzy network," *IEEE Trans. Industrial Electronics*, vol. 50, no. 6, pp.1305-1316, Dec. 2003.

[7] I. Ono and S. Kobayashi, "A real-coded genetic algorithm for function optimization using unimodal normal distribution crossover," in *Proc. 7th ICGA*, 1997, pp. 246-253.

[8] L.J. Eshelman and J.D. Schaffer, "Real-coded genetic algorithms and interval-schemata," *Foundations of Genetic Algorithms 2*, pp. 187-202, 1993.

[9] I. Daubechies, *Ten lectures on wavelets*. Philadelphia, PA: Society for Industrical and Applied Mathematics, 1992.

[10] X. Yao and Y. Liu, "Evolutionary programming made faster," *IEEE Trans. Evolutionary Computation*, vol. 3, no. 2, pp. 82-102, July 1999.

[11] X. Yao, "Evolving artificial networks," *Proceedings of the IEEE*, vol. 87, no. 7, pp. 1423-1447, 1999.

TABLE II. SIMULATION RESULTS FOR f_1 TO f_6

f_1 ($\times 10^{-4}$), number of iteration: 500				
	AveBXover+WM	AveBXover+NUM	UNDXBXover+WM	UNDXBXover+NUM
Mean	**1.6139**	4.1263	4.0840	12.614
Best	**0.00048**	0.0205	0.6095	4.5084
Std Dev	**2.9147**	5.1841	4.0513	6.7184
t-test ([AveBXover+WM]–[UNDXBXover+NUM]) = −10.62				
f_2 ($\times 10^{-2}$), number of iteration: 500				
	AveBXover+WM	AveBXover+NUM	UNDXBXover+WM	UNDXBXover+NUM
Mean	**0.24781**	0.38064	2768.9	2785.1
Best	0.02396	**0.01624**	2660.0	2638.0
Std Dev	**0.16678**	0.35854	59.999	62.845
t-test ([AveBXover+WM]–[UNDXBXover+NUM]) = −313.3				
f_3 ($\times 10^{0}$), number of iteration: 200				
	AveBXover+WM	AveBXover+NUM	UNDXBXover+WM	UNDXBXover+NUM
Mean	**0**	**0**	10.180	12.640
Best	**0**	**0**	1.0000	3.0000
Std Dev	**0**	**0**	5.0130	5.6524
t-test ([AveBXover+WM]–[UNDXBXover+NUM]) = −15.81				
f_4 ($\times 10^{0}$), number of iteration: 100				
	AveBXover+WM	AveBXover+NUM	UNDXBXover+WM	UNDXBXover+NUM
Mean	**−0.9721**	−0.9549	−0.8679	−0.8365
Best	**−1.0000**	−0.9999	−1.0000	−1.0000
Std Dev	**0.0530**	0.1198	0.3277	0.3689
t-test ([AveBXover+WM]–[UNDXBXover+NUM]) = −2.57				
f_5 ($\times 10^{-4}$), number of iteration: 200				
	AveBXover+WM	AveBXover+NUM	UNDXBXover+WM	UNDXBXover+NUM
Mean	**5.9125**	6.3380	17.863	29.451
Best	**3.1002**	3.3428	3.3147	5.1236
Std Dev	2.7085	**2.6445**	36.986	5.5539
t-test ([AveBXover+WM]–[UNDXBXover+NUM]) = −26.94				
f_6 ($\times 10^{-5}$), number of iteration: 500				
	AveBXover+WM	AveBXover+NUM	UNDXBXover+WM	UNDXBXover+NUM
Mean	**6.3267**	109.63	1174.3	1797.4
Best	**0.1832**	5.0020	1.7688	6.0472
Std Dev	**6.1773**	81.434	5773.2	6998.2
t-test ([AveBXover+WM]–[UNDXBXover+NUM]) = −1.81				

TABLE III. SIMULATION RESULTS FOR EXAMPLE OF ASSOCIATIVE MEMORY

MSE ($\times 10^{-4}$), number of iteration: 2000				
	AveBXover+WM	AveBXover+NUM	UNDXBXover+WM	UNDXBXover+NUM
Mean	**3.3939**	4.7965	7.4178	9.5312
Best	**2.5154**	2.8983	4.1872	5.8292
Std Dev	**0.4728**	0.9566	2.1383	1.6943
t-test ([AveBXover+WM]–[UNDXBXover+NUM]) = −24.67				