A New Nonlinear Genetic Algorithm for Numerical Optimization

ZHI-HUA CUI, JIAN-CHAO ZENG, YU-Bin Xu

(Division of system simulation and computer application,
Taiyuan Heavy Machinery Institute, Shanxi, P.R. China, 030024)
E_mail: cui_zhihua@sohu.com

Abstract:
Through mechanism analysis of simple genetic algorithm (SGA), every genetic operator can be considered as a linear transform. So some disadvantages of SGA may be solved if genetic operators are modified to nonlinear transforms. According to the above method, nonlinear genetic algorithm is introduced, and different nonlinear genetic operators with some probability are designed and applied to numerical optimization problems. The optimization computing of some examples is made to show that the new genetic algorithm is useful and simple.

Keywords:
Nonlinear Genetic Algorithm, Numerical Optimization, Nonlinear Genetic Operators

1. Introduction.

During the last three decades, there has been a growing interest in algorithms which rely on analogies to natural phenomena such as evolution, heredity, and immunity. The genetic algorithm (GA) belongs to one category of these best known algorithms, and done in the late 1960s and the early 1970s at the University of Michigan under the direction of J. Holland led to GA as it is known today.

Typically a GA is characterized by the following components:
- a genetic representation (or an encoding) for the feasible solutions to the optimization problem.
- a population of encoded solutions.
- a fitness function that evaluates the optimality of each solution.
- control parameters.

At the same time GAs’ disadvantages have appeared: 1) the calculating time is sometimes too long; 2) after a long time waiting for the calculations GAs cannot assure that the solution is optimal.

However some disadvantages of GAs has appeared such as premature convergence and calculation efficiency. To avoid these disadvantages some scientists have attempted to improve GAs in various ways. B. Sareni [1] used fitness sharing and niching methods to avoid premature convergence. S. Tsutsui et al. [2] introduced the concept of a bi-population scheme for real coded GAs (b-GAs) and the goal of b-GAs have some advantages in performing global exploration of the search space and avoiding being trapped at local optima. A lot of other scientists made attempt to improve GAs in other ways [3][4][5].

Through mechanism analysis of GA operators, every genetic operator can be considered as a linear transform. So, if we want to reduce the emergence of premature convergence and other faults, it is a useful method to modify the structure of genetic operators as nonlinear transform. By the first analysis several kinds of nonlinear crossover and mutation operators [6][7] are given. In this paper, by mechanism analysis of SGA, a new nonlinear genetic algorithm (NGA) which uses some different operators will be introduced.

In Section 2, a simple framework of NGA is given, including the form of nonlinear crossover and...
mutation operators, the structure of the nonlinear genetic algorithm; in Section 3, some experimental results about the comparison of SGA and NGA are given; in Section 4, we summarize the new nonlinear genetic algorithm, and give some future research aspects.

2. The new nonlinear genetic algorithm.

2.1 Mechanism analysis of GAs.

Without loss of generality, we will only deal with global minimization problems, which can be formalized as

$$\max F(x), x \in D = [a_1, b_1] \times \ldots \times [a_n, b_n]$$

From[3], we have drawn a conclusion that crossover, mutation, selection operators can be considered as a linear transform respectively, so if we modify the genetic operators to nonlinear transforms, then the effective of genetic algorithm will be better. Now, we’ll analyze the SGA, and give a new nonlinear genetic algorithm.

Suppose the current population is \( X(t) = (X_1^t, X_2^t, \ldots, X_n^t) \), where \( X_j^t \) is the chromosome, so the chromosomes can form a protruding polyhedron S. Let’s consider arithmetic crossover operator. First of all suppose \( x = (x_1, x_2, \ldots, x_n) \) and \( y = (y_1, y_2, \ldots, y_n) \) are the parent vectors, then the offspring \( w = (w_1, w_2, \ldots, w_n) \), \( v = (v_1, v_2, \ldots, v_n) \) are computed by

\[
w_j = \alpha x_j + (1 - \alpha) y_j \quad j = 1, 2, \ldots, n
\]

\[
v_j = \alpha y_j + (1 - \alpha) x_j
\]

where \( \alpha \) is a random number in [0,1] with uniform distribution. M(D) is the Lebesgue measure of domain area D, but the crossover search area is a line between point x and y, so the crossover operator only search within area S, and the probability of crossover is larger than mutation, so SGA can search S with a large probability, and this is the reason that SGA cannot assure that the solution is optimal. To solving this question, we give a new operator-dynamic mapping nonlinear crossover operator whose search area can go out of S with some probability. The ordinary mutation operator is randomly selecting a number with uniform distribution, of course the result can go out of S with probability \( (1 - \frac{m(S)}{m(D)}) \times p_m \), because of small mutation probability, this is the reason of premature convergence also, so if the new mutation operator can go out of S with a larger probability, then the number of premature convergence will decrease, and this is the new mutation operator.

2.2 Selection.

The NGA uses the truncation selection scheme adopted by BGA[1], which corresponds to the \((\mu+\lambda)\)-selection in ESs. The truncation selection only selects parents from the T% best individuals and the parents are mated randomly until the number of offspring is equal to the population size. The elitist
strategy (preserving the best individual found so far into the new generation) has also been used to get the best-so-far solution. Furthermore, self-mating is prohibited.

2.3 Dynamic mapping nonlinear crossover.

Let \( x = (x_1, x_2, ..., x_n) \) and \( y = (y_1, y_2, ..., y_n) \) be the parent vectors, the offspring vectors is \( w = (w_1, w_2, ..., w_n), v = (v_1, v_2, ..., v_n) \), now we will give a new method-dynamic mapping nonlinear crossover operator to calculate offspring vectors. First of all, \( [a_j, b_j] \) is the domain of \( j \)-th variable and divided into \( N \) parts indexed by \( \{U^j_1, U^j_2, ..., U^j_N\} \), and every part are the same Lebesgue measure. Suppose \( x_j \in U^j_{k_1}, y_j \in U^j_{k_2} \), and \( k_1 \leq k_2 \) (the same to \( k_1 > k_2 \) ), define the set \( \text{MiddlePart} = \{U^j_{k_1}, U^j_{k_2}, ..., U^j_N\} \), set \( \text{FirstPart} = \{U^j_1, ..., U^j_{k_1-1}\} \) and \( \text{LastPart} = \{U^j_{k_1+1}, ..., U^j_N\} \), then mapping \( g_1 \) and \( g_2 \) are defined to calculate the offspring vectors, where domain \( g_1 \) is the combination of FirstPart and MiddlePart with FirstPart shorten and MiddlePart enlarge, domain \( g_2 \) is the combination of MiddlePart and LastPart with MiddlePart enlarge and LastPart shorten. For instance, \( g_1 \) and \( g_2 \) can be defined as follows:

\[
g_1 : \text{FirstPart} \rightarrow \left[ \frac{1}{1 + \frac{m(U^j_s \setminus \text{FirstPart})}{2(b_j - a_j)}} \right]
\]

\[
\text{MiddlePart} \rightarrow [1,2]
\]

\[
g_2 : \text{MiddlePart} \rightarrow [2,1]
\]

\[
\text{LastPart} \rightarrow (1, \left[ \frac{1}{1 + \frac{m(U^j_s \setminus \text{LastPart})}{2(b_j - a_j)}} \right])
\]

where \( m(X) \) is the Lebesgue measure of set \( X \). Thus four temp individuals can be obtained by:

\[
z_1^j = g_1^{-1}(\alpha g_1(x_j) + (1 - \alpha)g_1(y_j))
\]

\[
z_2^j = g_1^{-1}(\alpha g_1(y_j) + (1 - \alpha)g_1(x_j))
\]

\[
z_3^j = g_2^{-1}(\beta g_2(x_j) + (1 - \beta)g_2(y_j))
\]

\[
z_4^j = g_2^{-1}(\beta g_2(y_j) + (1 - \beta)g_2(x_j))
\]
where $\alpha$ is a random number which made $z_1', z_2' \in [\frac{1}{1 + \frac{m(\bigcup_{x \in \text{FirstPart}} U^j_x)}{2(b_j - a_j)}}, 2]$, $\beta$ is a random number which made $z_3', z_4' \in [2, \frac{1}{1 + \frac{m(\bigcup_{x \in \text{LastPart}} U^j_x)}{2(b_j - a_j)}}]$. $g_1^{-1}$ is the inverse of mapping $g_1$, and the same to $g_2^{-1}$.

Thus four individuals are obtained and for convenience, suppose $F(z_1) \geq F(z_2) \geq F(z_3) \geq F(z_4)$, then offspring vectors are $z_1$ and $z_2$.

2.4 Dynamic mapping nonlinear mutation.

Let $x = (x_1, x_2, ..., x_n)$ be the parent vectors, the offspring vectors is $w = (w_1, w_2, ..., w_n)$, now we will give a new method-dynamic mapping nonlinear mutation operator. First of all, $[a_j, b_j]$ is the domain of $j$-th variable and divided into $N$ parts indexed by $\{U^j_1, U^j_2, ..., U^j_N\}$, and every part are the same Lebesgue measure. Suppose $x_j \in U^j_k$, define the set $\text{MiddlePart}= \{U^j_k\}$, set $\text{FirstPart}= \{U^j_1, ..., U^j_{k-1}\}$ and $\text{LastPart}= \{U^j_{k+1}, ..., U^j_N\}$, and mapping $g_1$ and $g_2$, where domain $g_1$ is the combination of FirstPart and MiddlePart with FirstPart enlarge and MiddlePart shorten, domain $g_2$ is the combination of MiddlePart and LastPart with MiddlePart shorten and LastPart enlarge. For instance, $g_1$ and $g_2$ can be defined as follows:

\[
g_1 : \text{MiddlePart} \rightarrow \left[ \frac{1}{1 + \frac{m(\bigcup_{x \in \text{FirstPart}} U^j_x)}{2(b_j - a_j)}} \right],
\]

\[
\text{FirstPart} \rightarrow [1,2]
\]

\[
g_2 : \text{LastPart} \rightarrow [2,1]
\]

\[
\text{MiddlePart} \rightarrow (1, \frac{1}{1 + \frac{m(\bigcup_{x \in \text{LastPart}} U^j_x)}{2(b_j - a_j)}}]
\]

where $m(X)$ is the Lebesgue measure of set $X$. Thus two temp individuals can be obtained by:

\[
z_1' = g_1^{-1}(\alpha g_1(x_j) + (1 - \alpha)g_1(x_j))
\]
\[ z'_2 = g_2^{-1}(\alpha g_2(x_j) + (1 - \alpha)g_2(x_j)) \]

where \( \alpha \) is a random number which made \( z'_2 \in [\frac{1}{m(\bigcup_{s \in \text{FirstPart}} U^{s}_{j,2})}, 2] \), \( \beta \) is a random number which made \( z'_2 \in [2, \frac{1}{m(\bigcup_{s \in \text{LastPart}} U^{s}_{j,2})}] \). \( g_1^{-1} \) is the inverse of mapping \( g_1 \), and the same to \( g_2^{-1} \). Thus two individuals are obtained and for convenience, suppose \( F(z'_1) \geq F(z'_2) \), then offspring vectors are \( z'_1 \) and \( z'_2 \).

The algorithm can be sketched as follows:

PROGRAM NGA
BEGIN
Initial the parameters: population size \( N \), nonlinear crossover probability \( P_c \), nonlinear mutation probability \( P_m \), initial population \( P(K), K := 0 \);
REPEAT
Perform dynamic mapping nonlinear crossover operator;
Perform dynamic mapping nonlinear mutation operator;
Perform selection operator;
\( K := K + 1 \);
UNTIL stopping criterion is satisfied
Output the best individual;
END.

3.1 Test Functions.
For the performance evaluation of the ANGA, we will use two functions They are usual test functions in global optimization.

-F1: Sphere Model,
\[ f_1(X) = \sum_{j=1}^{3} x_j^2, |x_j| \leq 5.12 \]

Its global minimum is at \((0,0)\) with \( F_1^* = 0 \).

-F2:
\[ f_2(X) = 5000 - 100(x_1 - x_2)^2 - (1 - x_1)^2, |x_j| \leq 2.048 \]

Its global minimum is at \((1,1)\) with \( F_2^* = 5000 \).

-F3: Schweel’s Problem 2.22
$$f_3(x) = \sum_{j=1}^{3} |x_j| + \prod_{j=1}^{3} |x_j| |x_j| \leq 10$$

Its global minimum is at $(0,0,0)$ with $F_3^* = 0$.

### 3.2 Experimental Results.

We implement our algorithm on PIV. In our experiments, we take the probability of recombination $P_c=0.85$, the probability of mutation $P_m=0.1$, and stop the computation when the expression

$$|F^* - F_{best}^*| < \varepsilon \cdot |F^*|$$

(if $F^* = 0$, it will be $|F^* - F_{best}^*| < \varepsilon$) is true. Here $F^*$ is the global optimum and $F_{best}^*$ denotes the function value of the best individual in current generation.

The experimental results are shown in Table 1. Each result was obtained through 50 random runs.

- $F_{eval}^*$ denotes the function evaluation number.
- $F_{per}^*$ denotes the function convergence radio.
- $X_{best}^*$ denotes the best value of individual by evaluation.
- $X_{worst}^*$ denotes the worst value of individual by evaluation.

<table>
<thead>
<tr>
<th>Function</th>
<th>Algorithm</th>
<th>error</th>
<th>$F_{per}$ (%)</th>
<th>$F_{eval}$</th>
<th>$X_{best}$</th>
<th>$X_{worst}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>SGA</td>
<td>0.1</td>
<td>100</td>
<td>896.96</td>
<td>0.012649</td>
<td>0.099284</td>
</tr>
<tr>
<td>F1</td>
<td>NGA</td>
<td>0.1</td>
<td>100</td>
<td>3.72</td>
<td>0.006159</td>
<td>0.098614</td>
</tr>
<tr>
<td>F1</td>
<td>SGA</td>
<td>0.01</td>
<td>20</td>
<td>4493.60</td>
<td>0.003712</td>
<td>0.008343</td>
</tr>
<tr>
<td>F1</td>
<td>NGA</td>
<td>0.01</td>
<td>100</td>
<td>138.08</td>
<td>0.000357</td>
<td>0.009795</td>
</tr>
<tr>
<td>F1</td>
<td>SGA</td>
<td>0.001</td>
<td>2</td>
<td>7176.0</td>
<td>0.000793</td>
<td>0.000793</td>
</tr>
<tr>
<td>F1</td>
<td>NGA</td>
<td>0.001</td>
<td>94</td>
<td>1856.5625</td>
<td>0.000059</td>
<td>0.000983</td>
</tr>
<tr>
<td>F2</td>
<td>SGA</td>
<td>0.1</td>
<td>100</td>
<td>54.24</td>
<td>0.001509</td>
<td>0.099280</td>
</tr>
<tr>
<td>F2</td>
<td>NGA</td>
<td>0.1</td>
<td>100</td>
<td>1.73</td>
<td>0.000049</td>
<td>0.067978</td>
</tr>
<tr>
<td>F2</td>
<td>SGA</td>
<td>0.01</td>
<td>100</td>
<td>585.1</td>
<td>0.000025</td>
<td>0.009776</td>
</tr>
<tr>
<td>F2</td>
<td>NGA</td>
<td>0.01</td>
<td>100</td>
<td>2.3</td>
<td>0.000028</td>
<td>0.008363</td>
</tr>
<tr>
<td>F2</td>
<td>SGA</td>
<td>0.001</td>
<td>80</td>
<td>3449.5</td>
<td>0.000023</td>
<td>0.000982</td>
</tr>
<tr>
<td>F2</td>
<td>NGA</td>
<td>0.001</td>
<td>100</td>
<td>3.0625</td>
<td>0.000003</td>
<td>0.000973</td>
</tr>
<tr>
<td>F2</td>
<td>SGA</td>
<td>0.0001</td>
<td>18</td>
<td>4926.33</td>
<td>0.000014</td>
<td>0.000099</td>
</tr>
<tr>
<td>F2</td>
<td>NGA</td>
<td>0.0001</td>
<td>100</td>
<td>5.4286</td>
<td>0.000003</td>
<td>0.000093</td>
</tr>
<tr>
<td>F3</td>
<td>SGA</td>
<td>0.1</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>F3</td>
<td>NGA</td>
<td>0.1</td>
<td>100</td>
<td>7.0</td>
<td>0.027720</td>
<td>0.098153</td>
</tr>
</tbody>
</table>

Table 1. Comparison of function $f_1(x,y)$, $f_2(x,y)$ and $f_3(x,y)$

### 4. Conclusion.

From the above Table, NGA is a better algorithm than SGA from evaluation number and convergence radio, it will bring some difficulty to the design of nonlinear genetic operators especially for the crossover operator, and the floating point representation makes the mutation operator instead of crossover operator the main searching operator.

Further research will include the design of other nonlinear genetic operators under our normalized representation and the implementation of distributed population models.
Acknowledgements

This work is supported by the Shanxi Natural Science Founds of P.R.China (Project No.20031041).

References