Proceedings of the 2003 IEEE
International Conference on Robotics,Intelligent Systems and Signal Processing
Changsha, China - October 2003

An Improved Fast-convergent Genetic Algorithm

WEI Gao[1,2]

(1 Wuhan Polytechnic University, Wuhan 430023 P. R. China)
(2 Logistical Engineering Institute, Chongqing 400041 P. R. China)
E-mail: wgaowh@etang.com

Abstract

As an effective global search method, genetic algorithm has been used in many engineering problems. When it is used in engineering, its slow convergence and poor stability have become the main problems. In order to overcome these problems, from the creation of the initial population, genetic operators, et al, an improved fast-convergent genetic algorithm is proposed. Through the simulation experiments of some hard-optimizing functions, the proposed algorithm shows its faster convergence and better stability than some existing algorithms'.

1 Introduction

Genetic algorithm (GA) recently has become a "hot spot" in the computer science, information science and artificial intelligence science [1-3]. It is a stochastic search algorithm based on Darwinian evolution and Mendelian genetic theories, proposed by American scholar Holland. Borrowing ideas from the species' evolution, genetic algorithm must encode the search problem, and every possible solution should be encoded to form a string, called chromosome. Create an initial population, and evaluate the population by a suitable fitness function. Then the genetic operators such as selection, crossover and mutation are implemented. The affection of the fitness function and selection operation mimic the natural selection. The other operators simulate the reproduction of the species. The selection operator selects or eliminates parents based on their fitness values. This operator guarantees the search direction. The crossover operator mimics the combination of genes, and guarantees the search range of the algorithm. While the mutation operator guarantees the global search.

From above principles, Goldberg summarized a general algorithm-Simple Genetic Algorithm (SGA). It has become the basis of the other algorithms. Figure 1 summarizes the working of the SGA, which has the following components:

① a population of binary strings,
② control parameters,
③ a fitness function,
④ genetic operators (crossover and mutation),
⑤ a selection mechanism,
⑥ a mechanism to encode the solutions as binary strings.

Simple Genetic Algorithm ()

```
{ initialize population;
  evaluate population;
  While termination criterion not reached
    { select solutions for next population;
      perform crossover and mutation;
      evaluate population;
    }
}
```

Figure 1 SGA structure

Compared with traditional optimizing methods, genetic algorithm has some advantages. First, it is a parallel algorithm, so the whole population is evaluated and operated. When evolution, all solutions in a population is searched simultaneously, which greatly decreases the probability that the algorithm is trapped in the local optimal solutions. Second, when using genetic algorithm in a real-world problem, the algorithm only requires the object function of the problem, and doesn't require other supplementary information. At last, it is a stochastic search method, not a determinate move. Which is a better simulation of the non-determination of nature' performance.

Because of the above advantages, genetic algorithm is widely used in the complicated engineering optimization problems [4-6]. But the slow-convergence and the instability of the algorithm from its random search have become the main problems in engineering usage. In order to overcome these problems, there have been a lot of studies and proposed many improved algorithms [7-11]. Although these studies have overcome the above problems at some extent, the results are not satisfied. For this reason, this paper proposed another improved fast-convergent genetic algorithm. This algorithm can overcome the above problems very well, which is verified by the simulated experiments.

This paper is organized as follows. In section II ,we discussed the basic principles of our algorithm and its basic methods. Section III describes the basic flow of the algorithm. In section Ⅳ, the simulated experiments are described and the results are compared with other algorithms'. The conclusions are presented in section Ⅴ.

2 The basic principles of the improved genetic algorithm

1. The creation of the initial population

The previous studies [5] show that the distribution of the initial population seriously affects the convergence of the algorithm. If the performance of the initial population is poor, the algorithm will have a slow convergence, even will not be convergent. The basic method to create the initial population is random creation in the solution space. This method has been used in many genetic algorithms. But this method does not consider the distribution of the initial population in the solution space. So many initial individuals may be put together in a small local space, which is bad for the extension of the search space and hard to find the global optimal solution. So it is necessary to study the creation of the initial population. In [8], a method to create the evenly distributed population based on the generalized Hamming distance between two initial individuals is proposed. But this method must calculate the generalized Hamming distance as each individual is created, which not only increases the calculating time but also makes operation inconvenient. Another method of artificial creation of the initial population is described in [5][10]. But this method requires knowing the nature of the problem so well that the good initial solutions could be given. This requirement is too strict to achieve in many cases. So this method decreases the robustness of the algorithm. In order to create good initial population quickly and not affect the other performance of the algorithm, this paper proposes a new method to create the initial population , called Small Region Creation Method (SRCM). In this method, the solution range of each parameter is divided into small regions. The number of the small regions is equal to population size. In each small region, an initial individual is created randomly. The SRCM can guarantee that there will be obvious difference between the initial individuals and the initial individuals distribute in the solution space evenly. So the algorithm based on this kind of initial population will be fast convergent and can find the global optimal solution easily.

2. Encoding mechanism

The convergence of the genetic algorithm has close relation with its encoding method [12]. Encoding method is the prime attention in design of the algorithm. Holland's Schema Theorem advocates the binary coding, gives the encoding rule of the minimum signs. Although binary coding is simple and easy to do, it will add extra-calculated time for encoding and decoding to the algorithm. Also, when encoding real numbers, the binary coding will generate the encoding error, which will decrease the precision of the algorithm. While real coding can overcome the above problems and can search larger space. Michalewicz [12] compared the affection of the two encoding methods through experiments, and revealed that the real coding is better. Antonise [13] disclosed Holland's error in giving the encoding rule of the minimum signs, and proves encoding method with more signs can be benefit to the algorithm's operation. So in our algorithm, the real coding is used.

3. Reproduction operation

In order to prevent the maximum individual from being destroyed by the genetic operators and preserve the good performance of the previous generation, our algorithm keeps the whole previous generation in the mating pool. This is a reproduction that allows parents to compete. This reproduction' affection on the algorithm has been researched specially [14], and it is called genetic algorithm with competition selection between adjacent two generations. Their research shows this reproduction is benefit to keep the population diversity, avoid the premature convergence very well and increase the efficiency and robustness of the algorithm.

4. Adaptive crossover of the optimum seeking from previous generation

This operation borrows ideas from the eugenics that the good offspring comes from the good parents. So, when the crossover is implemented, the two parents should be optimum seeking. The operation is carried out as follows, randomly select two individuals from the previous generation, and reserve the individual having the lager fitness value. (When the two have the equal fitness value, reserve one randomly.) And repeat the above operation once again. Then the two reserved individuals are crossed. The whole arithmetic crossover operator of the real coding is used [12]. The crossover process is as follows.
Supposing the two individuals that are to be crossed are $A=(a_1,a_2,\cdots a_i,\cdots,a_l)$ and $B=(b_1,b_2,\cdots,b_i,\cdots,b_l)$, then the offspring will be $A'=(a'_1,a'_2,\cdots a'_i,\cdots,a'_l)$ and $B'=(b'_1,b'_2,\cdots,b'_i,\cdots,b'_l)$.

where $a'_i = \alpha_i a_i + (1-\alpha_i) b_i$,

$b_i = \alpha_i b_i + (1-\alpha_i) a_i$.

(α_i is a random number in the range of [0,1])

In order to improve the performance of the crossover, adaptive crossover rate is applied [7]. So the evolutionary process controls the probability of crossover p_c. The expression of p_c takes the form

$$p_c = \begin{cases} (f_{\max} - f_c)/(f_{\max} - f_{avr}) , & \text{if } f_c \ge f_{avr} \\ 1.0 & \quad\;\; f_c < f_{avr} \end{cases}$$

where $f_{\max}$ is the maximum fitness value of the population,

f_{avr} is the average fitness value of the population,

f_c is the lager fitness value of the two solutions to be crossed.

5. Non-uniform adaptive mutation operation

Mutation operator makes the algorithm to search every point of the solution space. So the algorithm can find the global optimal solution. When using real coding, the mutation operator has become a very important searching operator, not only a simple process to keep the population diversity as it does in binary coding algorithm. Our algorithm applied the non-uniform mutation operator as it in [12]. Supposing the mutated individual is $A=(a_1,a_2,\cdots a_i,\cdots,a_l)$, where a_i is selected to be mutated. If the number range of the a_i is ($a_{i\max}$, $a_{i\min}$), after mutated, it will become

$$a_i' = \begin{cases} a_i + f(t, a_{i\max} - a_i) & \text{if } rad = 0 \\ a_i - f(t, a_i - a_{i\min}) , & \quad\;\; rad = 1 \end{cases}$$

where rad is a random number,

$f(t,y)$ is a function whose results are in [0,y], its expression is as follows

$$f(t,y) = y(1 - r^{(1-\frac{t}{T})^b})$$

where t^{*} is the number of the current evolutionary generation,

r is a random number in the range of [0,1],

T is the threshold value of the evolutionary generation,

B is a system parameter, which value generally equals to 2.

In order to improve the performance of the mutation, adaptive mutation rate is applied [7]. So the evolutionary process controls the probability of mutation p_m. The expression of p_m takes the form as follows:

$$p_m = \begin{cases} 0.5(f_{\max} - f_m)/(f_{\max} - f_{avr}) , & \text{if } f_c \ge f_{avr} \\ 0.5 & \quad\;\; f_c < f_{avr} \end{cases}$$

where $f_{\max}$ is the maximum fitness value of the population,

f_{avr} is the average fitness value of the population,

f_m is the fitness value of the individual to be mutated.

6. Selection operation

The follow two selection operations are used to generate the offspring from the mating pool.

① Elitist strategy

Find the individual with the maximum fitness value in the mating pool and preserve it in the offspring. So the individual with the maximum fitness value in the previous generation can be kept. For this reason, the algorithm will be convergent to the global optimal solution with the probability of 1.

② Tournament selection

Here the scale of the tournament is two. Select two individuals randomly from the mating pool, compare their fitness values and reserve the one with the larger fitness value in the offspring. Repeat this operation to generate the whole offspring population.

7. Termination criterion

Because the genetic algorithm is a stochastic search method, it is difficult to formally specify the convergence criteria. The termination criterion in this paper is that the difference of the maximum fitness value and the average fitness value is less than 10e-5. And in order to avoid infinite iteration, the number of maximum evolutionary generation is given as 200.

3 Process of the improved genetic algorithm

Based on the principles in above section, we have complied the program with computer language—Fortran 90. The flow chart of the program is as follows.

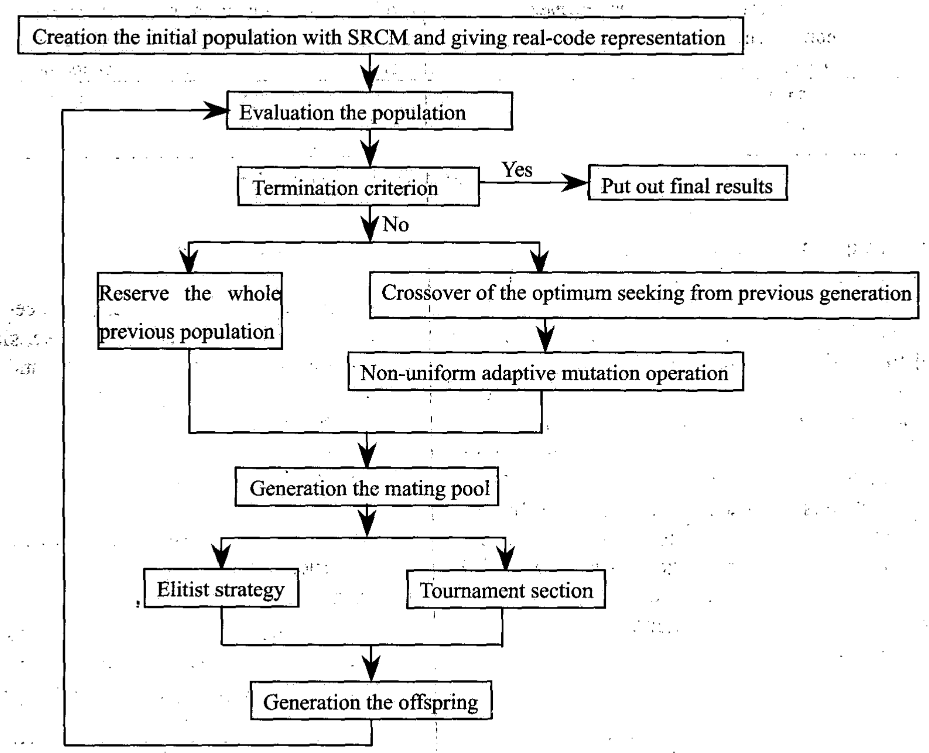
Creation the initial population with SRCM and giving real-code representation
Evaluation the population
Termination criterion
Yes
Put out final results
No
Reserve the whole previous population
Crossover of the optimum seeking from previous generation
Non-uniform adaptive mutation operation
Generation the mating pool
Elitist strategy
Tournament section
Generation the offspring

Figure 2 The flow chart of the improved genetic algorithm

Confined by the space, the whole program is not given here.

4 Simulation experiments

1 *First experiment*

In first experiment, we study the performance of the algorithm to solve the multi-modal optimization. Here we use Percy's function F1 and Schafer's function F2. All of the two functions have a lot of locally optimal solutions, and hard to be solved by traditional methods. The two functions are showed in table 1

Object functions	Range of the parameters	Threshold value of the optimization
F1: $f(x)=\frac{x_1^2+x_2^2}{2}+\cos(20\pi x_1)\cos(20\pi x_2)+2$	[-10,10]	3 (minimum)
F2: $f(x)=0.5-\frac{\sin^2\sqrt{x_1^2+x_2^2}-0.5}{[1-0.001(x_1^2+x_2^2)]^2}$	[-100,100]	1(maximum)

Table 1 Multi-modal object functions

Table 2 shows the results of our algorithm and other algorithms' to compare.

Object	Simple Genetic Algorithm	The algorithm in reference 10	Our algorithm

functions	Average number of convergent generation	Number of non-convergence	Average number of convergent generation	Number of non-convergence	Average number of convergent generation	Number of non-convergence
F1	81.9	15	51.7	6	11.2	2
F2	70.6	4	30.8	0	6.4	0

Table 2 Results of the experiments

In simulation experiments, the control parameters of SGA are N=100, P_c=0.65, P_m=0.01; Parameters in [10] are N=50, P_c=0.65, P_m=0.15; In our algorithm, the control parameter is N=50. In experiments, the function is optimized for 50 trails in each time. The average number of generation for attaining a solution with a fitness value equal to the threshold value is the criterion to judge the speed of the algorithm. If the algorithm do not attain a solution with a fitness value equal to the threshold value for 200 generations in each trail, we call this trail non-convergence. The number of the non-convergence in 50 trails is the criterion to judge the stability of the algorithm. As show in table 2, the convergent speed and stability of our algorithm is very good and works well in multi-modal optimization.

2 *Second experiment*

In order to study the convergent speed of our algorithm, in this experiment we use the simple, single hump function (Dejong's first function) to study and compare the results with many existing algorithms'.

The function is as follows

$$f(x_i) = 100 - \sum_{i=1}^{3} x_i^2$$

where x_i (i=1~3) are the optimizing parameters, their numerical range is as follows

$x_i \in [-5.12, 5.12]$

It is a problem to search the maximum result. The maximum result of the function is 100 at (0,0,0).

The results of the experiment is showed in table 3

Threshold value of the function	99	99.5	99.9	99.95	99.99	99.995	99.999	99.9995
SGA	3	8	30	45	185	—	—	—
Uniformly exchanged GA	3	8	54	90	157	—	—	—
GA using migration	4	7	40	78	179	—	—	—
A fast convergent GA	3	7	26	58	134	163	—	—
Algorithm in [8]	2.63	4.47	8.4	11.63	38.83	46.1	51.27	52.87
Our algorithm	1	1	1	1	1.73	1.87	2.4	3.2

Table 3 Comparison of the average number of generation for attaining a solution with fitness value equal to the threshold value by different algorithm

Notes: the average number of the generations is the average number for 30 trails; the non-convergence is indicated by —; the other algorithms in table 3 can be found in the references of reference [8].

From table 3, we can find that the convergent speed of our algorithm is very fast. Even when the threshold value is equal to 100, the average number of generations is only 6.68. So our algorithm is a very fast and good algorithm, and it is better than many existing algorithms.

5 Conclusion

From creation of evenly distributed initial population, application of adaptive genetic operators, competition between adjacent two generations and elitist strategy, our algorithm improves the genetic algorithm. Through experiments, we can conclude that our algorithm improves the convergent speed and the stability of the genetic algorithm very well, and its performance is better than many existing algorithms'.

References

[1] M. Srinivas, L. M. Patnaik, "Genetic Algorithm: A Survey", *IEEE computer*, vol. 27, no. 6, pp. 17-26, 1994.

[2] D. B. Fogel, "An Introduction to Simulated Evolutionary Optimization", *IEEE Trans. On SMC.*, vol. 24, no. 1, pp. 3-14, 1994.

[3] W. Atmar, "Notes on the Simulation of Evolution", *IEEE Trans. On SMC.*, vol. 24, no. 1, pp. 130-147, 1994.

[4] Gao Wei, Zheng Yingren, "Back Analysis of Rock Mass Parameters Based on Evolutionary Algorithm", *Journal of Hydraulic Engineering*, no. 8, pp. 1-5, 2000.

[5] A. M. Zalzala, P. J. Fleming, *Genetic Algorithms in Engineering Systems,* The Institution of Electrical engineers, 1997.

[6] Zhou Ming, Sun Shudong, *Genetic Algorithms: Thoery and applications,* China Defense Industrial Press, 1999.

[7] M. Srinivas, L. M. Patnaik, "Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms", *IEEE Trans. On SMC.*, vol. 24, no. 4, pp. 656-667, 1994.

[8] Wu Bin, Wu Jian, and Tu Xuyan, "Research of Fast Genetic Algorithm", *Journal of UEST of China*, vol. 28, no. 1, pp. 49-53, 1999.

[9] J. C. Potts, T. D. Giddens, and S. B. Yadav, "The Development and Evaluation of an Improved Genetic Algorithm Based on Migration and Artificial Selection", *IEEE Trans. On SMC.*, vol. 24, no. 1, pp. 73-86, 1994.

[10] Han Wanlin, Zhang Youdi, "Improvement of Genetic Algorithm", *Journal of CUMT*, vol. 29, no. 1, pp. 102-105, 2000.

[11] J. A. Millar, W. D. Potter, "An evaluation of Local Improvements Operators for Genetic Algorithms", *IEEE Trans. On SMC.*, vol. 23, no. 5, pp. 1340-1351, 1993.

[12] Z. Michalewicz, *Genetic Algorithms + Data Structure = Evolution Programs,* Springer-Verlag, 1996.

[13] J. Antonise, "A New Interpretation of Schema Notation That Overturns the Binary Encoding Constrain", in *Proc. Of the 3rd Int. Conf. On GA*, 1989, pp. 86~91.

[14] Yu Haibin, Wang Haobo, and Xu Xinhe, "A Genetic Algorithm with Competition Selection Between Adjacent Two Generations and Its Applications to TSP", *Information and Control*, vol. 29, no. 4, pp. 309-311, 2000.