An Improved Genetic Algorithm Based Complex-valued Encoding

Yan Wang[†] ,Shangce Gao[†], Huiran Zhang[†] and Zheng Tang[††],

Graduate School of Innovative Life Science, University of Toyama, Toyama, Japan

Summary
Genetic algorithm is a useful tool to tackle optimization problems. In this paper the complex numbers were introduced into the traditional genetic algorithm, in which binary or real value data representation was used in the past, and a complex-value encoding genetic algorithm was proposed. Comparing with the conventional genetic algorithm which based on real-valued encoding or binary encoding, the proposed algorithm expands the dimension for search region and avoided getting into the local minimum efficiently. The computer simulation results showed it is useful for the problem of function optimization.

Key words:
complex number, complex-valued encoding, genetic algorithms, diploid, allele

1. Introduction

Genetic algorithms (GAs), originally conceived by Holland [1], represent a fairly abstract model of Darwinian evolution and biological genetics. They evolve a population of competing individuals (The individuals are represented as genes on a chromosome) using fitness-biased selection, random mating, and a gene-level representation of individuals together with simple genetic operators (typically, crossover and mutation) for modeling inheritance of traits. These GAs have been successfully applied to a wide variety of problems including numerical function optimization [2], image processing [3-4], combinatorial optimization [5-6], machine learning [7], and so on.

. In simple genetic algorithm (SGA) which was proposed by Holland, the chromosomes were expressed by binary-coded，and the algorithm was used to solve the optimization problem of discrete variables. However, with the change of problem, solution precisions of binary-encoded are restriction by the length of chromosomes. If the length is too short, the quality of optimal solution would be influenced, and if the length is too long, search space would be increased, the efficiency would be reduced. Therefore, decimal-encoding was introduced to express chromosomes. Later, Gray coded, real coded and sign coding [8] were also used in GA according to the different problems. That is to say, the coding method is not fixed, and it can be changed with different problems and applications. Inspired by the idea of complex encoding which was used in artificial neural network [9-11], we extended the usual real-valued GA to complex numbers domain. Based on diploid genotypes, we proposed a new complex-encoding genetic algorithm and used proposed algorithm in the question of function optimization. As 2D message can be expressed by one complex number, variables of the same quantity can save the information two times more than real-encoding variables. This property provided the encoding method of GA for a new idea. It can be certified by simulation results that the proposed algorithm mined the individual diversity of the population, and overcame the shortcoming that GA pre-matured convergence easily, and helped to improve the search efficiency of GA. Moreover, because expressional form and operators of complex number had diversity, GA encoded by complex number can use more flexible operator than the GA which were encoded by binary or real number. This is a new idea of GA in solution practical application problem in real world.

We arranged the rest of the paper as follows: Section 2 described the structures of the proposed GA. In section 3, the experimental setup was explained. Results were presented and comparisons with real-coded GA were made in section 4. Last, we presented conclusions in section 5.

2. Proposed complex genetic algorithm

In recent years, genetic algorithm which was based diploid genotypes (DGA) was catching more and more attention of scientists [12]. DGA had better properties of search ability and compute speed than GA based haploid (HGA). It is more important that DGA had expended bigger express space than HGA. In the nature it is usually that double-chain or multi-chain chromosomes are used by complex biological system. For diploid, in the process of reproduction, two chromosomes which came from father and mother respectively compose a chromosome pairs. The chromosome pair has the properties that come from parents. As complex code has the feature of 2directions, we thought to realize the proposed complex-encoded GA by DGA. In the following, we would explain the algorithm structure what we proposed.

Manuscript received June 5, 2010
Manuscript revised June 20, 2010

2.1 Chromosomes Structure

In diploid, the individuals of population were composed by two gene chains of the same length. Every gene chain had two alleles. In the proposed algorithm, we used complex number to express the allele of chromosome pair, corresponding to the real part and imaginary part, which were called real gene and imaginary gene respectively.

For a question with m-independent variables, let there are m complex numbers $z_k = x_k + iy_k$ (k=1, 2… m), denoting as:

$$((x_1, x_2, \ldots, x_m), (y_1, y_2, \ldots, y_m))$$

Compose the chromosomal two-chain structure, if let:

$$r = (x_1, x_2, \ldots, x_m), d = (y_1, y_2, \ldots, y_m)$$

then, r is real gene string and d is imaginary gene string.

2.2 Initial Population of Random Individuals

According to the definition region of problem $[a_k, b_k]$(k=1, 2 … m), M modulus of complex numbers were produced: $\rho_k \in \left[0, \frac{b_k - a_k}{2}\right]$, k=1,2..m. While m arguments of complex numbers were produced: $\theta_k \in [0, 2\pi]$, k=1,2..m. The m complex numbers can be expressed by the form of polar coordinate as follow:

$$x_k + iy_k = \rho_k(\cos\theta_k + i\sin\theta_k), (k=1,2..m) \quad (1)$$

According to the method which described above, generate a number of individuals

$$\left((\rho_1, \rho_2, \ldots, \rho_m), (\theta_1, \theta_2, \ldots, \theta_m)\right)$$

to formation a initial group.

2.3 Selection

Selection is the stage of a genetic algorithm in which individual genomes with better adaptabilities were chosen from a population for later breeding (recombination or crossover). The main purpose of select options was to improve the global convergence and computational efficiency. But a small proportion of less fit solutions were selected. This helped keep the diversity of the population large, preventing premature convergence on poor solutions. In this paper, we used elitist selection [13] and roulette-wheel selection [14], that is to say, retaining the best individuals in a generation unchanged in the next generation, and the other individuals of population were chosen by fitness proportionate selection.

2.4 Crossover

Crossover options specified how the genetic algorithm combined two individuals, or parents, to form a crossover child for the next generation. The crossover probability determined the probability of an individual to be selected for crossover.

Crossover operator was an important method which can obtain new individuals and it was a key to deciding if an algorithm had the convergence property. In order to improve the speed of search, in the algorithm we proposed, arithmetic crossover [15] operator were used in real-part and imaginary-part respectively. That was to say, not only the length of modulus but the angle of arguments were changed in every crossover operator.

(a) real-part's crossover

Let the two modulus component of the parents individuals which were selected to crossover are:

$$\rho_1 = (\rho_1^1, \rho_1^2, \ldots, \rho_1^m)$$

$$\rho_2 = (\rho_2^1, \rho_2^2, \ldots, \rho_2^m)$$

it would generate a new individual ρ_c as follow:

$$\rho_c = (\rho_c^1, \rho_c^2, \ldots, \rho_c^m)$$

a random number γ ($\gamma \in [0,1]$) was generated by system, then,

$$\rho_c^k = \gamma\rho_1^k + (1-\gamma)\rho_2^k \text{ (k=1, 2…m.)} \quad (2)$$

(b) imaginary-part's crossover

Supposed that the two argument components of the parent individuals which were selected to crossover are:

$$\theta_1 = (\theta_1^1, \theta_1^2, \ldots, \theta_1^m)$$

$$\theta_2 = (\theta_2^1, \theta_2^2, \ldots, \theta_2^m)$$

and the argument component of offspring is:

$$\theta_c = (\theta_c^1, \ldots, \theta_c^k, \ldots, \theta_c^k) \text{(k=1, 2…m)}$$

here

$$\theta_c^k = \gamma\theta_1^k + (1-\gamma)\theta_2^k \quad (3)$$

here γ is a random value between 0 and 1.

It can be proved simply that the angle which gained by crossover still belongs $[0, 2\pi]$.

2.5. Mutation

Mutation was also a good method for obtaining new individual. Mutation options specified how the genetic algorithm made small random changes in the individuals in the population to create mutation children. Mutation enabled the genetic algorithm to search a broader space and provided genetic diversity from one generation of a population of algorithm chromosomes to the next, which can prevent the phenomenon of premature. It was more important in searching action than other algorithms that the complex-encoding had an inherent property of population diversity. In this paper we used two kinds of mutation operations: adaptive mutation [16] for modulus and Muhlenbein mutation [17] form argument.

(a) adaptive mutation for real-part

Set $(\rho_1, \rho_2, \ldots, \rho_m)$ is a modulus component of a complex number string, ρ_k is selected to mutation, and definition region of ρ_k is $[0, \frac{b_k - a_k}{2}]$.the modulus component after mutated is:

$(\rho_1, \ldots, \rho_{k-1}, \rho_k^*, \ldots, \rho_m)$

there,

$$\rho_k^* = \begin{cases} \rho_k + \Delta\left(T, \frac{b_k - a_k}{2} - \rho_k\right), & \text{if rand} \leq 0.5 \\ \rho_k - \Delta(T, \rho_k), & \text{if rand} > 0.5 \end{cases} \quad (4)$$

Where, rand is a random number between 0 and 1, which generate by system. Function $\Delta(T, y) = y \cdot (1 - r^{T^\lambda})$, there, r is also a random number between 0 and 1. λ is a parameter which determine the degree of non-compatible, and it plays a role to adjust the local search area, its value can be 2 to 5. T is a mutation temperature, defined as:

$$T = 1 - \frac{f(z)}{f_{max}} \quad (5)$$

where, f(z) is the fitness values of individual, f_{max} is the top limit of f(z), usually used the maximum value of current population. If the modulus mutated is large than $\frac{b_k - a_k}{2}$, the mutation would be canceled.

(b) Muhlenbein mutation for imaginary-part

The mutation operator of imaginary-part was adjusted on the basis of Muhlenbein mutation. Let $(\theta_1, \theta_2, \ldots, \theta_m)$ is an argument component of a complex number string. If θ_k was selected to mutate, then the argument component after mutated is $(\theta_1, \ldots, \theta_{k-1}, \theta_k^*, \ldots, \theta_m)$, where $\theta_k^* = \theta_k \pm A\delta$, $A\delta$ gives the range of mutation, taking "+" or "-" be the same probability, A=0.2π, $\delta = \sum_{j=0}^{T} \alpha_j \cdot 10^{-j}$, the mutation precision τ is a positive integer which can be determined by the precision we want to acquire, $\alpha_j \in \{0,1\}$, it takes 1 by the probability of $\frac{1}{\tau+1}$. That is to say, $P(\alpha_j = 1) = \frac{1}{\tau+1}$.

2.6 Termination Criteria

This generational process is repeated until a termination condition has been reached. The maximum number of generations must be defined together with the desired fitness level. By satisfying either one of the above criteria, the GA will terminate.

3. Process for GAs based on complex genetic algorithm

1) In the beginning, two populations with the size of N chromosomes $(\rho_1, \rho_2, \ldots, \rho_m)$ and $(\theta_1, \theta_2, \ldots, \theta_m)$ were created randomly by system, which ρ_k and θ_k indicate the modulus and angle of complex of allele respectively. The chromosomes' length is m. ($\rho_k \in [0, \frac{b_k - a_k}{2}]$, $\theta_k \in [0, 2\pi]$, k=1, 2…m). The 2*N chromosomes contained the initial population with N chromosomes. Then the variable x_k which corresponded by allele can be expressed as follows:

$$x_k = \rho_k \cos\theta_k + \frac{a_k + b_k}{2} \quad (6)$$

there k=1, 2…m;

2) Evaluates the fitness of each individual in that population;

3) If the pre-specified the termination criteria are reached, then stop;

4) Select the best-fit individuals for reproduction;

5) Breed new individuals through crossover and mutation operations to give birth to offspring;

6) Go back to step 2.

4. Simulation experiment and results

When tackling a real world problem, the conventional approach is to test first the algorithm on a set of analytical functions. Most real world applications involve objective functions considerably more expensive to evaluate than an analytical one. Consequently, it is usually unviable to test the effectiveness of an algorithm directly on the real problem. There is the assumption that the chosen sets of functions share some characteristics with the target problem. Based on this assumption, one applies the algorithm that performed well on the benchmark in the expectation that it will do also well on the real world problem [18].

The function was selected for several reasons. First, it has been widely used, which will allow an extensive comparison with previously published algorithms. Also, the function has a number of features that are known to be hard for optimization algorithms and believed to be present in many real world problems. The Ackley function was a continuous, multimodal function obtained by modulating an exponential function with a cosine wave of moderate amplitude. Its topology is characterized by an almost flat (due to the dominating exponential) outer region and a central hole or peak where the modulations by the cosine wave became more and more influential. The

basin of these local minima increased in size as one moved away from the global minimum, as discussed by [19]. Thus, this function would be useful to study the behaviors of the algorithms when initialized in a highly multimodal region.

For comparing the searching capabilities of algorithm which we proposed with the traditional real-coded genetic algorithms, we chose Ackley function as the optimization problem. The formula of the Ackley function was expressed as follow [20]:

$$\min f(x1, x2) = -20 \exp\left(-0.2\sqrt{\frac{1}{n}\sum_{j=1}^{n} x_j^2}\right) - \exp\left(\frac{1}{n}\sum_{j=1}^{N}\cos(2\pi x_j)\right) + 20 + e \quad (7)$$

there $-5 \le x_j \le 5$, j=1, 2, and n=2. Figure 1 is the visualization of Ackley's path function.

4.1 The process of experiments

1) Program language: Matlab was used in simulation experiments.

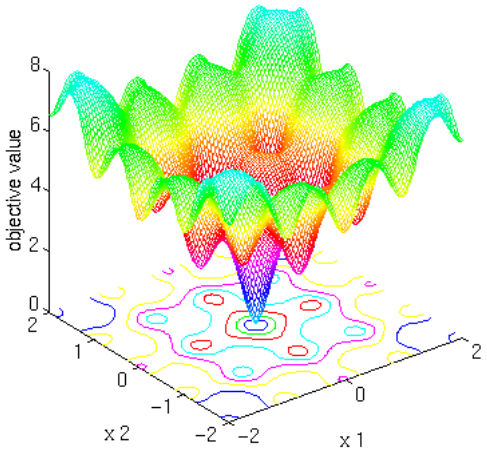


Fig1: The Visualization of Ackley's Path Function. Focus around the area of the global optimum at [0,0] in an area from -2 to2.

2) Fitness function

$$\text{fitness}(x_1, x_2) = \frac{1}{\sqrt{f(x_1, x_2)}} \quad (7)$$

Because Ackley function was a continuous real function, in order to achieve the fitness of individual, complex number had to be change to real number by formula(6).

3) Genetic operations and parameter setting

In simulation, complex-encoded GA which we proposed used genetic operators which had been described in section 2, and real-encoded GA's genetic operators were on the whole the same: selection operator used the combination of elite selection and roulette, crossover operator used arithmetic crossover and mutation operator used Muhlenbein mutation. For comparing the two algorithms scientific, fairly, the same parameters were set in the experiments: population size N=100, probability of crossover $p_c = 0.5$, mutation precision τ was 15, adaptive mutation λ =2, f_{max} was the largest value of current fitness.. In elite algorithm, the number of best individuals which were retained in each generation was 2.

4.2 Simulation results

4.2.1 The comparison of evolution generation

At the first experiment, we compared both algorithms by evolution generation. The experiment was begun with the same initial population. And the target minimum was 1e-10. Figure 2 showed the comparison of the complex-encoded GA and real-encoded GA in evolution ability. (a) was generation from 0 to 70, (b-d) was the details of (a) that the generation from 0 to 10, 11 to 35, and 36 to 70 respectively. From figure 2, we can see that at the beginning (generation 0 to 13), the both algorithm convergence quick, and at the generation 14, the real-encoded GA meet a local minimum and premature convergence. But for complex-encoded GA, when it met a local minimum, the value of modulus and argument changed continuously and jumped out the local minimum at generation 16. From generation 13, the real-encoded GA still stayed at the local minimum and cannot change. While complex-encoded GA reached the global minimum at generation 69.

4.2.2 The comparison of success rate

When monitoring the performance we also measured of success rate (percentage of cases when an optimum was found). We did two kinds of experiments of success rate. 1e-6 and 1e-15 were selected as the target minimum value. When the minimum is 1e-6, the two algorithms were executed two kinds of situations by the maximum generation being 50 and 100. When the target value was 1e-15, the algorithms were executed two kinds of situations by the maximum generation being 200 and 1000. All the programs were executed 1000 times respectively, and the average results were recorded in table 1. From the table 1, we can see that:

i) When evolution generation were changed

No matter the optimum value was 1e-6 or 1e-15, when the value of N was changed from 50 to 100 or from 200 to 1000, the success rate of real-encoded GA hardly changed, (The success rate of real-coded GA was changed from 33.5% to 34% in the experiment that the target was 1e-6, and from 6.5% to 7% in the experiment when the target was 1e-15.) But for the complex-encoded GA which we proposed the changing were obviously. (The success rate of complex-coded GA was changed from 85.6% to 100%

when the target was 1e-6 and from 39.8% to 74% when the target was 1e-15.)

ii) When the precision of optimum values were improved

From the table 1, we also can see that when the precision of optimum values were improved from 1e-6 to 1e-15, the success rate of real-encoded GA descended from over 30% to below 10% when the generation is enough, while the success rates of complex-encoded GA still kept over 70%.

4.2.3 The comparison of the optimal value which algorithm can reach

At the experiment 3, we tested the optimal value which the both algorithms can reach. We executed both algorithms by the same initial population. The maximum generation was 1000. The terminal condition was that the minimum value was not changed in 100 generation continuous or the maximum generation was reached. The experiments also were done 100 times respectively, and the best solution and the worst solution were recorded in table 2.

From the table 2, we can see that the both algorithms can reach the best solution of 8.88e-16. (This is the minimum value which can be computed by Matlab with the data type of long float, the decimal part is 50 bit, and the minimum value is 2^(-50)=8.88e-16.) But for the worst solution, the real-encoded GA's expression was awful. It only found the 0.12966 after 107 generation were evaluated. While for the complex-encoded GA the worst solution was 7.99e-15. By the way, the real-encoded GA can reach the ideal value 4 times of the programs were executed by 100 times and the complex-encoded GA achieved the ideal value 48 times of 100 times.

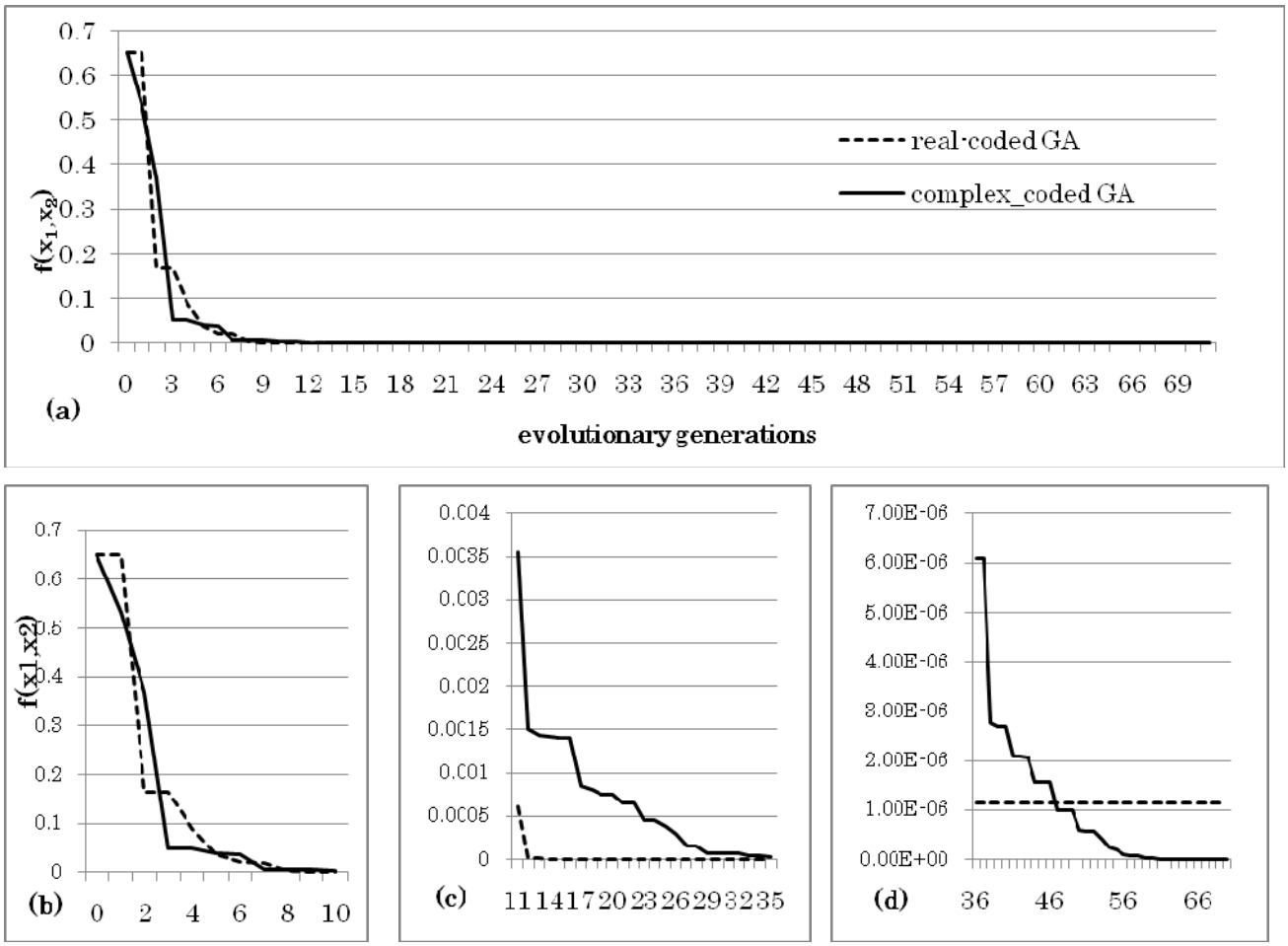


Fig 2: Comparison of the complex-encoded GA and real-encoded GA in evolution ability by function value which generation was from 0 to 70.

Table 1: Simulation Results: Performance comparison between real-coded GA and Complex-coded GA

		Real-encoded GA	Complex-encoded GA
fval<10^{-6}	N=50	33.5%	85.6%
	N=100	34%	100%
fval<10^{-15}	N=200	6.5%	39.8%.
	N=1000	7%	74%

Table2: Simulation Results: the best solution and the worst we can achieve by both algorithms

	The best solution			The worst solution		
	X(1)	X(2)	fval	X(1)	X(2)	Fval
Real-encoded GA	-7.07e-18	1.68e-16	8.88e-16	0.192875	0.0003745	0.064443
Complex-encoded GA	-3.65e-18	9.93e-17	8.88e-16	-1.82e-15	2.32e-15	7.99e-15

5. Conclusions

We presented a GA (complex-coded GA) which had been shown to be effective in optimizing high dimensional real-variable functions. Complex-coded GA's performance has been tested on Ackley function, in which the function had a number of features that is known to be hard for optimization algorithms and believed to be present in many real world problems.

In this paper, we described the genetic operator based on complex encoding by introduced diploid. Complex numbers have the character of 2D, which made complex number express more and more information than binary or real number. In addition, when the 2D's complex numbers were mapped into one direction, that is to say that when 2 dimensional space encoding were corresponded to one dimensional encoding space, there was no doubt that we can improve the diversity of population. Compared with the traditional GA, complex-encoded GA did not perform as well as this algorithm for problems with few local minima. This algorithm not only had the inherent ability to maintain the population diversity, improve the search ability for global optimal point, but also provides a useful exploration for the development of GA. Simulation results show its effectiveness.

References

[1] J.H.Holland, Adaptation in Natural and Artificial Systems, Ann Arbor: The University of Michigan Press, 1975

[2] K.DeJong, "The Analysis and behavior of a Class of Genetic Adaptive Systems." PhD thesis, University of Michigan, 1975.

[3] D.E.Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning", Addison-Wesley, Reading, MA, 1989

[4] C.Caldwell and V.S.Johnston, "Tracking a criminall suspect through "face-space" with a genetic algorithm." In R.K. Belew and L.B.Booker, editors, Processdings of the Fourth International Conference on Genetic Algorithms, pages416-421, Morgan Kaufmann, 1991

[5] M.Gorges-Schleuter. "ASPARAGOS: an asynchronous parallel genetic optimization strategy." In J.D. Schaffer, editor, Proceedings of the Third International Conference on Genetic Algorithms, pages 422-427. Morgan Kaufmann,1989

[6] S.Bagchi, S.Uckun, Y.Miyabe, and K.Kawamura. "Exploring problem-specific operators for job shop scheduleing." In R.K. Bele and L.B.Booker, editor, Proceedings of Fourth International Conference on Genetic Algorithms, pages 10-17. Morgan Kaufmann, 1991

[7] S.Forrest and G.Mayer-Kress. "Genetic algorithms, nonlinear dynamical systems, and models of international security." In L.Davis, editor, Handbook of Genetic Algorithms, chapter 13, pages 166-185. Van Nostrand Reinhold, 1991

[8] http://en.wikipedia.org/wiki/Genetic_algorithm

[9] G.M. Georgiou, C.Koutsougeras, "Complex domain backpropagation", IEEE Trans. Circuits System-II: Analog Digital Signal Process. 39(5) (1992) 330-334.

[10] T.Nitta, T.Furuya, "A complex back-propagation learning", Trans. Inform. Process. Soc. Japan 32(10) (1991) 1319-1329 (in Japanese)

[11] Sanjay S.P. Rattan and William W. Hsieh. "Complex-valued neural network for nonlinear complex principal component analysis", Neural Networks, vol18, issue 1, January 2005, pages 61-69

[12] Goldberg, D.E., &smith, R.E. (1987). "Nonstationary function optimization using genetic algorithms with dominance and diploidy", Proceedings of the Second International Conference on Genetic Algorithms, 59-68

[13] Z.Tang, Y.Zhu, G.Wei, J.Zhu. "An Elitist Selection Adaptive Genetic Algorithm for Resource Allocation in Multiuser Packet-based OFDM Systems", Journal of Communications. Vol. 3, No. 3, July 2008. 27-32

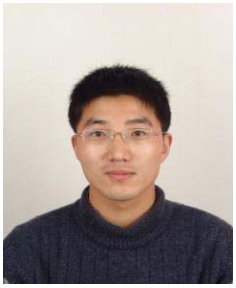
[14] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," in Foundations of Genetic Algorithms. San Mateo, CA: Morgan Kaufmann, 1991, pp. 69–93.

[15] C.R.Houck, J.A.Joines, and M.G.Kay. "A genetic algorithm for function optimization: a Matlab implementation", NCSU-IE Technical Report 95-09, North Carolina Satate University, 1995

[16] Culotta, Elizabeth, "A Boost for 'Adaptive' Mutation", Science, 265:318, 1994

[17] Muhlenbein,H. "How genetic algorithms really work I: Mutation and hillclimbing", In R.Manner&B.Manderick (Eds.), Parallel problem solving from nature, 2 (pp. 15-25). Amsterdam: Elsevier.
[18] Ballester, P.J., Ballester, P.J., Tavassoli, Z., King, P.R., "Our calibrated model has no predictive value: An example from the petroleum industry." In Pamee, I.C., ed.: Proceedings of the Adaptive Computing in Design and Manufacture VI. (2004) In Press.
[19] Chellapilla, K., Fogel, D.B.: Fitness distributions in evolutionary computation: Analysis of local extrema in the continuous domain. In Angeline, P.J., Michalewicz, Z., Schoenauer, M., Yao, X., Zalzala, A., eds.: Proceedings of the Congress of Evolutionary Computation. Volume 1., Mayflower Hotel, Washington D.C., USA, IEEE Press(1999) 760-767
[20] D. H. Ackley. "A connectionist machine for genetic hill-climbing", Boston: Kluwer Academic Publishers, 1987.

Yan Wang received the B.S. degree from Jinzhou teacher college, Jinzhou, China in 2002 and M.S. degree from Bohai University, Jinzhou, China in 2008. Now she is working toward the Ph.D. degree at University of Toyama, Toyama, Japan. Her main research interests are genetic algorithm, neural network, optimization algorithms and pattern recognitions.



Shangce Gao received the B.S. degree from Southeast University, Nanjing, China in 2005. Now, he is working toward the Ph.D. degree at Toyama University, Toyama, Japan. His main research interests are multiple-valued logic, artificial immune system and artificial neural system.

Huiran Zhang received the B.S. degree from Shanghai University, Shanghai, China in 2005 and the M.S. degree from University of Toyama, Toyama, Japan in 2008. From 2008, he is working toward the Ph. D at University of Toyama, Toyama, Japan. His current research interests are the intellectual information technology and neural networks.

Zheng Tang received the B.S. degree from Zhejiang University, Zhejiang, China in 1982 and an M.S. degree and a D.E. degree from Tshinghua University, Beijing, China in 1984 and 1988, respectively. From 1998 to 1989, he was an Instructor in the Institute of Microelectronics at Tshinhua University. From 1990 to 1999, he was an Associate Professor in the Department of Electrical and Electronic Engineering, Miyazaki University, Miyazaki, Japan. In 2000, he joined University of Toyama, Toyama, Japan, where he is currently a Professor in the Department of Intellectual Information Systems. His current research interests included intellectual information technology, neural networks, and optimizations.