GARS: An Improved Genetic Algorithm with Reserve Selection for Global Optimization

Yang Chen [*]
Graduate School of Information, Production and Systems, Waseda University
2-7 Hibikino, Wakamatsu, Kitakyushu, Fukuoka
808-0135, Japan
yangchen@ruri.waseda.jp

Jinglu Hu [†]
Graduate School of Information, Production and Systems, Waseda University
2-7 Hibikino, Wakamatsu, Kitakyushu, Fukuoka
808-0135, Japan
jinglu@waseda.jp

Kotaro Hirasawa [‡]
Graduate School of Information, Production and Systems, Waseda University
2-7 Hibikino, Wakamatsu, Kitakyushu, Fukuoka
808-0135, Japan
hirasawa@waseda.jp

Songnian Yu [§]
School of Computer Engineering and Science, Shanghai University
149 Yanchang Road, Zhabei District, Shanghai 200072, China
snyu@staff.shu.edu.cn

ABSTRACT

This paper investigates how genetic algorithms (GAs) can be improved to solve large-scale and complex problems more efficiently. First of all, we review premature convergence, one of the challenges confronted with when applying GAs to real-world problems. Next, some of the methods now available to prevent premature convergence and their intrinsic defects are discussed. A qualitative analysis is then done on the cause of premature convergence that is the loss of building blocks hosted in less-fit individuals during the course of evolution. Thus, we propose a new improver - GAs with Reserve Selection (GARS), where a reserved area is set up to save potential building blocks and a selection mechanism based on individual uniqueness is employed to activate the potentials. Finally, case studies are done in a few standard problems well known in the literature, where the experimental results demonstrate the effectiveness and robustness of GARS in suppressing premature convergence, and also an enhancement is found in global optimization capacity.

[*]Graduate Student, Waseda University.

[†]Associate Professor, Waseda University.

[‡]Professor, Waseda University.

[§]Professor, Shanghai University.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*; G.1.6 [**Numerical Analysis**]: Optimization—*Global optimization*

General Terms

Algorithms

Keywords

Building block hypothesis, evolutionary computation, genetic algorithms, population diversity, premature convergence, reserve selection

1. INTRODUCTION

Genetic algorithms (GAs) are search and optimization algorithms based on the principles of natural evolution, which were first introduced by Holland [9]. An overview about GAs and their implementation in various fields was given by Goldberg [7] or Michalewicz [11].

The important characteristics of GAs are: (i) use a code of parameters, (ii) work on a population of points, (iii) use probabilistic evolution rules to obtain a satisfying result, and (iv) use simple operations on the value of the objective function [1]. Hence, GAs have been successfully applied to many fields such as the combinatorial optimization in recent decades, especially in searching for the global optimum of complex problems with many local optima, where traditional optimization methods may fail to provide reliable results effectively.

In applying GAs to solve large-scale and complex real-world problems, however, confronted with the conflict between high accuracy and low time consumption, GAs often result in an unsatisfactory compromise, characterized by a lack of accuracy and a slow convergence, when a more precise solution is expected within the given time. Furthermore, one of the commonest difficulties frequently encountered is *premature convergence* [3, 6].

In this work, we develop a new mechanism called *reserve selection* to enhance the performance of GAs, aiming to avoid premature convergence, i.e. maintain population diversity, and finally achieve global optimization.

2. RELATED WORKS

Roughly speaking, premature convergence occurs when the population in a genetic algorithm is trapped in such a suboptimal state that most of the genetic operators can no longer produce offspring that outperforms their parents [6].

It is widely recognized that the decrease of population diversity leads directly to premature convergence. One intuitive explanation may be that individual solutions are selected through a fitness-based process, where fitter solutions (as measured by a *fitness function*) are typically more likely to be selected. Popular and well-studied selection methods such as *roulette wheel selection* and *tournament selection* are stochastic and designed so that a small proportion of less fit solutions are also selected. This, to some extent, helps keep the population diversity high, preventing premature convergence on poor solutions.

Moreover, several methods have been proposed to fight against premature convergence in GAs [2, 5, 7, 8]. For example, the enlargement of population size, the increase of mutation rate, the use of more disruptive crossover operators, and the modification of fitness assignment, for example, will reduce the possibility of losing genes and thus maintain population diversity.

However, enlarging population size which is a simple and intuitive way will bring about a remarkable growth in both time and space complexity. Besides, a few individuals being mutated may not resume the diversity of the whole population since the new offspring will soon be cleared away with large possibility. It can be extremely hard to jump out of the suboptimum by mutation around the current optimal individual. As for the scaling of fitness function, it is shown that it does not affect the converging speed obviously [10].

3. GENETIC ALGORITHMS WITH RESERVE SELECTION (GARS)

3.1 Premature Convergence

As we all know, when premature convergence happens, all the individuals in a population tend to be identical with almost the same fitness value. It is really hard for such a low-entropy population without sufficient genetic information to evolve any more. However, how does the abundant gene contained in the initial population which is usually randomized with a certain size lose gradually with the evolution process?

During the course of evolution, the individuals with higher fitness values are selected more than once, whereas many less-fit individuals are rejected. However, the well-known *building block hypothesis* suggests that short, low-order, and highly-fit schemata (or building blocks) contribute to the evolution towards better solutions [7]. Note that it is still possible for the individuals which seem poor at the moment to contain building blocks, which even though not having been "activated"yet, may play an important role in future evolution. Unfortunately, they are permanently eliminated together with their host individuals from the population, subject to the law of "survival of the fittest".

Table 1: Great loss of genetic information in evolution

Generation ID	N_u	N	R_{loss}
1	68	100	68%
2	65	100	65%
3	69	100	69%
4	66	100	66%
5	70	100	70%
6	66	100	66%
7	64	100	64%
8	72	100	72%
9	67	100	67%
10	65	100	65%

The great gene loss in evolution for the multiple sequence alignment problem (see also Section 4.3) is shown in Table 1, where a tournament selection is employed with *tournament size* 5. We measure the loss of genetic information by the *loss rate* (R_{loss}), calculated using the ratio of the number of unselected individuals in each generation (N_u) to the population size (N) as follows.

$$R_{loss} = \frac{N_u}{N} \times 100\%,\ 0 \le N_u \le N$$

The result obtained reveals two facts: (1) In each generation, only about one-third of the individuals are selected to produce offspring, while all the others are died out due to their low fitness. Apparently, the genetic information has not been fully utilized, which seems a disadvantage to global optimization; (2) Besides, there may exists some redundancy in offspring, since they derive from only a small part of their ancestors. This may indirectly give rise to premature convergence.

As a matter of fact, the building blocks buried in less-fit individuals should be reserved, partially or as much as possible, being expected to be unleashed in a new environment, i.e. a new host individual. Meanwhile, the offspring redundancy should be effectively removed, without sacrificing the converging speed excessively. In the solution given below, with population size remaining unchanged, we do a good tradeoff between the two objectives.

3.2 Reserve Selection

In this section, we present an improver of conventional GAs, which is a new selection mechanism called *reserve selection.* As we stated before, we hope it can hit the target effectively - to prevent premature convergence for the purpose of global optimization.

The new algorithm is based on the technique called *population segmentation*, which divides the offspring population into two parts: *non-reserved area* and *reserved area*, as displayed in Figure 1.

- **Non-reserved area (NRA):** similar to the population of standard GAs, this part mainly works as an intensified searcher, approaching the local optima via the *exploitation* of the individuals of good quality. The

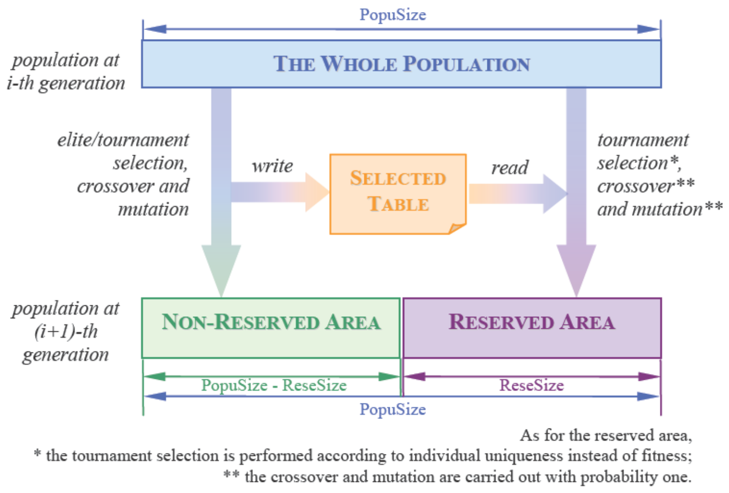

PopuSize
population at i-th generation
THE WHOLE POPULATION
elite/tournament selection, crossover and mutation
write
SELECTED TABLE
read
tournament selection*, crossover** and mutation**
population at (i+1)-th generation
NON-RESERVED AREA
RESERVED AREA
PopuSize - ReseSize
ReseSize
PopuSize
As for the reserved area,
* the tournament selection is performed according to individual uniqueness instead of fitness;
** the crossover and mutation are carried out with probability one.

Figure 1: Reserve selection via population segmentation

offspring in NRA usually originate from more-fit individuals of the preceding population since they are produced via elite and tournament selection. Considering that NRA is now merely a part of the whole population, the redundancy in offspring could to some extent be alleviated when compared with the conventional method.

- **Reserved area (RA):** this part is specially set up to reserve the building blocks buried in poor individuals and thus maintain a diversified search so as to *explore* the global optimum. The size of RA is called *reserve size*. The offspring in RA derive exclusively from those not having been selected yet in producing NRA, which are of low fitness values in general. In doing so, a record called *selected table* is used to label the individuals selected in producing NRA, with $O(N)$ space complexity where N is population size. RA helps to restrict the loss of genetic information.

The way of segmentation, or the choice of reserve size, depends on various demands and thus a balance can be done between exploitation and exploration. In particular, when reserve size becomes zero, the implementation completely reduces to a conventional GA.

3.3 Evolution of Reserved Area

It is far from satisfactory just reserving less-fit individuals, because the building blocks hosted there still remain to be activated. Therefore, proper genetic operators need to be performed to turn the reserved genes to full account.

While the same procedure of selection and recombination is applied for NRA as that in standard GAs, a set of genetic operators is elaborately designed for RA. Considering that individuals available for producing RA are in common of low fitness values, we cancel the elite selection, and both crossover rate and mutation rate are set to one, inducing potential building blocks to migrate into a better environment as soon as possible. In addition, we renovate the tournament selection by substituting the *uniqueness* of individuals for *fitness value* as the criterion of selection so as to diversify the population.

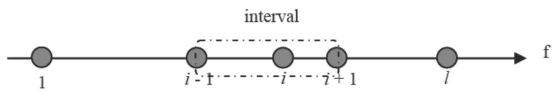


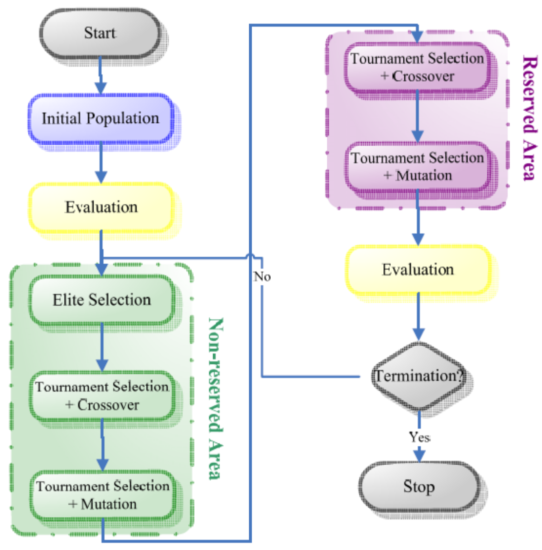
Figure 2: Estimate the density of individuals

The uniqueness is defined using the *crowded comparison* approach originally presented in NSGA-II [4]. A slight modification is done in order to match it with single-objective problems. We get an estimate of the density of individuals surrounding a particular individual (i) in the population by calculating the distance between two points $(i-1$ and $i+1)$ on both sides of this point along the axis of fitness value (f), which is shown in Figure 2. This quantity $d(i)$, called the *crowding distance*, serves as a measure of the size of the largest interval enclosing the point i without including any other point in the population.

The computation of crowding distance requires sorting of the population according to the fitness value in their ascending order of magnitude. Thereafter, the boundary individuals (individuals with smallest and largest fitness values) are assigned an infinite distance value. All other intermediate individuals are assigned a distance value equal to the absolute difference in the fitness values of its two adjacent individuals. The following algorithm clearly outlines the crowding distance computation procedure of all individuals in a population P.

```
ALGORITHM 1. Compute the crowding distance
crowding-distance-assignment(P)
$l = |P|$                      number of individuals in P
$P' = sort(P, f)$              sort P by fitness f
$d(1) = d(l) = \infty$         boundary points are always selected
for $i = 2$ to $(l-1)$         for all other points
      $d(i) = |f(i+1) - f(i-1)|$
```

Here, $f(i)$ refers to the fitness value of the i-th individual. The above algorithm has $O(N \log N)$ time complexity, where N is population size. After each individual i in the population P is assigned a distance metric $d(i)$, we can compare



Start
Initial Population
Evaluation
Elite Selection
Tournament Selection + Crossover
Tournament Selection + Mutation
Non-reserved Area
No
Tournament Selection + Crossover
Tournament Selection + Mutation
Reserved Area
Evaluation
Termination?
Yes
Stop

Figure 3: Flowchart of GAs with Reserve Selection

the *uniqueness* of individuals, $u(i)$, which is a monotonic increasing function of $d(i)$. Between two different individuals, the one with a higher uniqueness is more preferred in selection procedure.

For practical use, according to the genetic operators through which the offspring is reproduced, we can further divide NRA into three subareas: $NRA.elite$, $NRA.crossover$ and $NRA.mutation$, while RA may consist of two subareas: $RA.crossover$ and $RA.mutation$. The flowchart of Genetic Algorithms with Reserve Selection (GARS) is displayed in Figure 3. The difference from conventional GAs lies in its two-stage procedure, where the evolution of RA is carried out successively after that of NRA. Note that the individuals selected in producing some subarea of NRA are urged to be reused for others in order to derive more benefit from superior genes. However, this rule is not applicable to RA so that more building blocks could be reserved.

4. CASE STUDIES AND EXPERIMENTAL RESULTS

4.1 Multimodal Function Optimization

Compared with monomodal function optimization, the multimodal function optimization is more influenced by premature convergence. Once the process gets caught in a local optimal state, one will probably receives solutions far from the global optimum.

In this test, we compare the new algorithm with the conventional GA to evaluate its performance, where the real number encoding is employed. The population size is set to 10, and the generation number is set to 100. The reserve size is set to 4. Besides, our results are the average outcome of 1000 independent runs.

1. Search the minimum of function

$$f(x) = (x+0.9)(x+0.7)(x+0.2)(x-0.4)$$
$$(x-0.7)(x-0.9)+0.04,\ x\in[-1,+1]$$

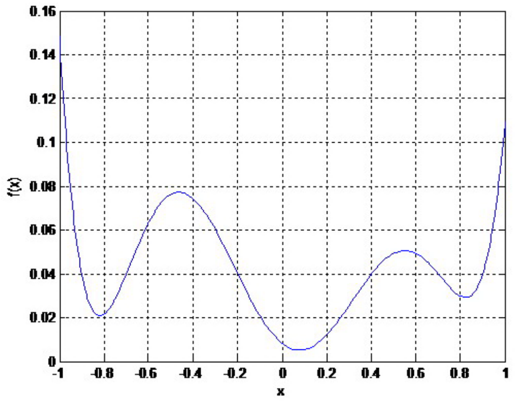


Figure 4: $f(x)$ **on** $[-1,+1]$

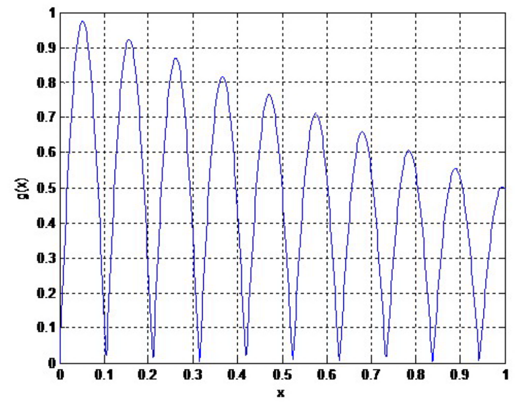


Figure 5: $g(x)$ **on** $[0,1]$

The function curve is depicted in Figure 4, where there are three minima and the global one is

$$x = 0.077152,\ f_{min}(x) = 0.00517542$$

2. Search the maximum of function

$$g(x) = |\sin(30x)| \times (1-\frac{x}{2}),\ x\in[0,1]$$

which has many maxima as depicted in Figure 5. The global one is

$$x = 0.0517900,\ g_{max}(x) = 0.9739626$$

The results are shown in Table 2. Given the same population size (space limit) and generation number (time limit), the precision of our new algorithm (1.69% and 0.40%) is much higher than that of the conventional GA (14.14% and 7.79%). In fact, premature convergence occurs in conventional GA during the search period and the population can no longer be renewed in that all individuals tend to be identical. The proposed method takes the advantage of the reserve selection to prevent GAs from premature convergence such that the better solution can always be searched for.

4.2 Traveling Salesman Problem

The traveling salesman problem (TSP) is a problem in discrete or combinatorial optimization, which belongs to the

Table 2: Compare GA and GARS in multimodal function optimization

Function	Global Optimum	GA	GARS
$f(x)$	0.00517542	0.00590726	0.00526281
$g(x)$	0.9739626	0.898096	0.970057

Table 3: Compare GA and GARS in traveling salesman problem

Instance	TSP Dimension	GA	GARS
tsp225	225	28434.8	25044.5
kroA200	200	264647	227692
ch130	130	35133.9	29614.2

class of problems known as NP-complete. One of the traditional lines on attacking the NP-hard problems is devising "suboptimal"or heuristic algorithms. TSP is a touchstone for many general heuristics devised for combinatorial optimization such as genetic algorithms. That is why here we study in this case the proposed method.

We also adopt a real number encoding scheme in the experiment. Both the population size and the generation number are set to 100. The reserve size is set to 30. In addition, the results are the average outcome of 100 independent runs. The testing instances are all picked from TSPLIB, a library of 110 sample instances of traveling salesman problem (and related problems) from various sources and of various types for the benchmark of TSP algorithms.

Table 3 lists the solutions to three instances in TSPLIB, obtained using both conventional GA and GA with Reserve Selection. Evidently, our improved algorithm performs better than the conventional one with a promotion in result fitness by 11.92%, 13.96% and 15.71% respectively.

More details are given by fitness curves for the instance *kroA200* in Figure 6, contrasting the evolution process of both methods. In the earlier period of evolution, the propose method converges more slowly than the conventional one till the $23rd$ generation since building blocks are being accumulated but their powers have not been fully released yet. In the long run, with the building blocks saved by the reserve selection mechanism being activated in new environments, our algorithm progressively demonstrates its efficiency, while the conventional method loses its convergence speed resulting from lack of population diversity.

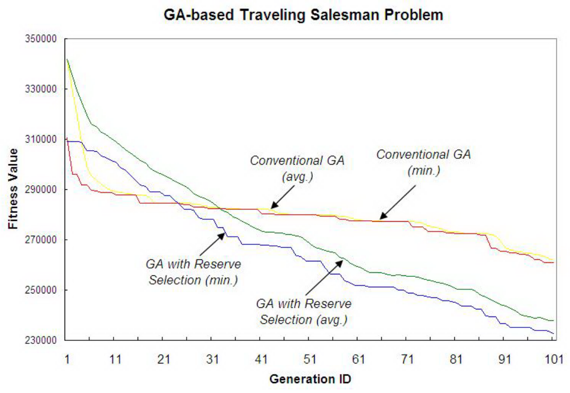


Figure 6: Fitness curves of traveling salesman problem

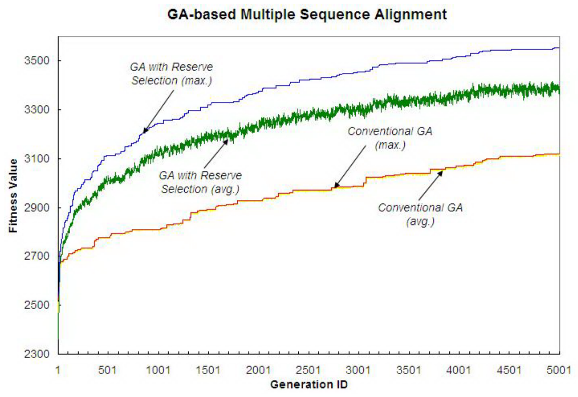


Figure 7: Fitness curves of multiple sequence alignment

4.3 Multiple Sequence Alignment

The multiple sequence alignment (MSA) is among the most important and most challenging tasks in Bioinformatics, which is crucial to humans' understanding of the phenomena of life. However, the problem is characterized by very high computational complexity in terms of both time and space. In fact, MSA with the sum-of-pairs (SP) score is an NP-hard problem [15]. Most practical algorithms are therefore based on heuristics producing quasi-optimal alignments. The problem of aligning multiple sequences can be converted into that of searching for optimal solutions in a problem space. Hence, a genetic algorithm can be designed for MSA.

In this case, we also make a comparison of both algorithms to show the effectiveness of the new improver, where binary coding is used. The population size and the generation number are set to 100 and 5000 respectively. The reserve size is set to 30. The biological sequences we use for the experiment are retrieved from the SRS database [12], whose general information is listed in Table 4.

Figure 7 portrays the evolution process of both methods. As can be learned, on the one hand, our new algorithm converges more quickly than a conventional GA as the evolution process passes by so that a better alignment can be found in a shorter period of time. On the other hand, for the conventional method, there is an undistinguishable difference between the maximal and average fitness values which is used as a yardstick to measure premature convergence by Srinivas and Patnaik [14]. However, the proposed method maintains population diversity successfully by reserving the building blocks hosted in less-fit individuals from the very beginning, as indicated by the different gene loss before and after reservation in Table 5.

5. CONCLUSIONS AND FUTURE WORK

Premature convergence is one of the major problems faced by genetic algorithms. The population diversity gets worse and worse since the building blocks buried in less-fit individuals are losing along with the evolution. GARS, an improved genetic algorithm with a reserve selection mechanism, saves the potential building blocks to a reserved area

Table 4: Multiple sequences to be aligned

ID	Entry Name	Molecule Type	Sequence Length
1	EMBLRELEASE: AB075784	genomic DNA	500
2	EMBLRELEASE: AB080848	genomic DNA	500
3	EMBLRELEASE: AB099717	genomic DNA	500

Table 5: Gene loss in evolution before and after reservation

Generation ID	N_u Before Reserve	R_{loss} Before Reserve	N_u After Reserve	R_{loss} After Reserve
1	61	61%	30	30%
2	68	68%	37	37%
3	67	67%	36	36%
4	66	66%	35	35%
5	67	67%	36	36%
6	69	69%	38	38%
7	64	64%	33	33%
8	66	66%	35	35%
9	67	67%	36	36%
10	64	64%	33	33%

in each offspring population, and attempts to activate them with the help of innovative genetic operators. The experiments are conducted in several cases, whose results indicate that the proposed method succeeds in avoiding premature convergence by maintaining a diverse population. Moreover, compared with standard GAs, the improver exceeds in both solution quality and convergence speed, without a notable increase in the computational complexity. We believe this idea may be generalized to some population-based evolutionary algorithms other than GAs.

Future work will mainly involve four aspects. First, we will examine by experiments how different reserve sizes affect the performance; second, the working principle of building blocks will be investigated to explain in theory why GARS works with a quantitative analysis; thirdly, we will try combining GARS with other effective methods to build a hybrid solution; fourthly, the proposed method will be applied to solve a wider range of complex problems. Our goal is to build a highly efficient and robust evolutionary algorithm.

6. REFERENCES

[1] J. Andre, P. Siarry, and T. Dognon. An improvement of the standard genetic algorithm fighting premature convergence in continuous optimization. *Advances in Engineering Software*, 32:49–60, 2001.

[2] H. G. Cobb and J. J. Grefenstette. Genetic algorithms for tracking changing environments. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 523–530, San Mateo, CA, 1993. Morgan Kaufmann.

[3] L. Davis, editor. *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, New York, 1991.

[4] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002.

[5] K. A. De Jong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems.* Doctoral thesis, Department of Computer and Communication Sciences, University of Michigan, Ann Arbor, 1975.

[6] D. B. Fogel. An introduction to simulated evolutionary optimization. *IEEE Transactions on Neural Networks*, 5(1):3–14, January 1994.

[7] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison Wesley Longman, 1989.

[8] J. J. Grefenstette. Genetic algorithms for changing environments. In R. Manner and B. Manderick, editors, *Parallel Problem Solving from Nature 2*, pages 137–144, Amsterdam, North Holland, 1992.

[9] J. H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, MI, 1975.

[10] J. Ma and et al. The great mutation used to improve the searching quality of GA. *Control Theory and Applications*, 15(3):404–407, 1998. In Chinese.

[11] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag, Berlin Heidelberg New York, 1996.

[12] NIAS DNA Bank SRS database. Website http://srs.dna.affrc.go.jp/srs8/.

[13] G. Reinelt. TSPLIB - a traveling salesman problem library. *ORSA J. Comput.*, 3:376–384, 1991.

[14] M. Srinivas and L. M. Patnaik. Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, 24(4):656–667, April 1994.

[15] L. Wang and T. Jiang. On the complexity of multiple sequence alignment. *Journal of Computational Biology*, 1(4):337–348, 1994.