Memoirs of the Faculty of Engineering, Okayama University, Vol.30, No. 1, pp.111-120, December 1995

Genetic Algorithm with Evolutionary Chain-Based Mutation and Its Applications

Ju Ye,* Masahiro Tanaka** and Tetsuzo Tanino**

(Received October 2 , 1995)

Mutation is one of the important operators in genetic algorithm. In traditional genetic algorithm, mutation is activated stochastically. In this way it is unknown and cannot be controlled for which individuals to be mutated. Therefore, it is unavoidable that some good individuals are destroyed by mutation and then the evolutionary efficiency of the genetic algorithm is dampened. Owing to this kind of destructivity of mutation, the operator of mutation has to be limited within a very small probability, and the potentiality of mutation is consequently limited. In this paper, we present an evolutionary chain-based mutation and a control strategy of reasonable competition, in which the heuristic information provided by the evaluation function is well utilized. This method avoids the blindness of stochastic mutation. The performance improved in this method is shown by two examples, a fuzzy modeling for the identification of a nonlinear function and a typical combinatorial optimization problem—the traveling salesman problem.

1 INTRODUCTION

Widely used in a number of applications to find optima in very large search spaces[1,2,3,4,5], genetic algorithms were first developed by Holland in 1975[6] as an attractive class of heuristic computation models inspired by the Darwinian natural evolutionary theory. They simulate the evolution of natural populations according to the principles of natural selection and "survival of the fittest", encode a potential solution to a specific problem on a simple chromosome-like data structure and apply recombination operators to these structures so as to preserve critical information. An implementation of a genetic algorithm begins with a population of (typically random) chromosomes, evaluates these structures and allocates reproductive opportunities in such a way that those chromosomes which represent better solutions to the target problem are given more chances to "reproduce" than those chromosomes which are poorer solutions. A canonical genetic algorithm[7] (CGA) can be expressed as follows:

Step 1. Initialization: generate initial population P of binary coded individuals at random, and set $N := |P|$ (the size of P);

Step 2. Evaluation: evaluate all individuals in P using fitness function, and let a temporary population $P' := \phi$ (an empty set);

Step 3. Reproduction: copy individuals several times from P and generate a parent set S_p with the probability which is proportional to the fitness by using a roulette wheel;

Step 4. Crossover: pick up two individuals from S_p according to a crossover probability p_c, mate the two parents with one-point crossover, two-point crossover, or uniform crossover to form two offsprings, insert the offsprings into P',and repeat step 4 until all parents in S_p are picked up;

*Graduate School of Natural Science and Technology
**Department of Information Technology

Step 5. Mutation: alter each bit of offsprings in P' with a mutation probability p_m, then go to step 2 until the termination criterion holds.

As genetic algorithms are typically concerned with solving complex optimization problems, the problem of genetic algorithm's evolutionary efficiency has been attracting the attention of genetic algorithm community[8]. Mutation, as one of the main operators in genetic algorithms, is responsible for re-introducing inadvertently "lost" gene values (alleles), providing some new starts for search, and preventing premature convergence of the algorithms. It is an operator not to be ignored and a far more profound operator than has ever been recognized[9]. However, in traditional genetic algorithm, mutation is activated stochastically. In this way it is unknown and cannot be controlled for which individuals in a population to be mutated, so it is unavoidable that some good individuals, i. e. the individuals that are promising to generate good offsprings, are destroyed by the mutation operator and then the evolutionary efficiency of the genetic algorithm is dampened. Owing to this kind of destructivity of mutation, the operator of mutation has to be limited within a very small probability, and the potentiality of mutation is consequently limited.

Moreover, as the evaluation function is the link between the genetic algorithm and the problem to be solved, the evaluation function provides the heuristic information for evolutionary search. Therefore, how to use this kind of heuristic information (in the present and past) is influential in the efficiency of evolutionary search.

In section 2 an evolutionary chain-based mutation is described, in which the heuristic information provided by the evaluation function is well utilized for mutation, and the blindness of stochastic mutation and the disruptive problem of mutation are avoided. In section 3, a control strategy of reasonable competition is proposed, which brings the effects of crossover and mutation into full play. The genetic algorithm with this method is described in section 4. In section 5, the improvement of evolutionary efficiency through of this method is shown by two examples, a fuzzy modeling for the identification of a nonlinear function and a typical combinatorial optimization problem—the traveling salesman problem.

2 EVOLUTIONARY CHAIN-BASED MUTATION

The evaluation function plays the same role in genetic algorithm that the environment plays in the natural evolution. The evaluation function provides the heuristic information for evolutionary search, and how to use this kind of heuristic information (present and past) is influential in the efficiency of evolutionary search. In genetic algorithm, mutation is applied to each child of the current population after crossover. It randomly alters each gene with a small probability. It results in some individuals being changed and some individuals keeping the same. The operation is in effect equivalent to

1) determining some individuals from the current population with a new probability, and then

2) altering randomly some genes of each individual determined in 1).

We consider that the key of avoiding disruptive problem of mutation is how to determine the individuals to be mutated. If we have a method of only mutating the unpromising (to generate good offsprings) individuals, then the disruptive problem of mutation can be avoided.

One basis for predicting the future is to examine the past. Here we present a method of evolutionary chain-based mutation, which utilizes the evaluation values on the evolutionary chain of an individual, as the heuristic information, to examine the past and predict the future of the individual.

First let us define some terms. An individual is referred to as "the first generation individual", if it is in the initial population or it is just generated by mutation. An individual x is referred to as an individual z's "representative parent" or "father", if z is generated by crossing x over another individual y and x is better than y, or if z is a copy of x from the last population, and the link from z to x is referred to as z's "father pointer". An individual is referred to as "the i-th generation individual", if its father is the $(i-1)$-th generation individual, where $i > 1$. An evolutionary chain of x is a link of x with the father pointer from x to the first generation individual.

Evolutionary chain-based mutation means determining an individual to be mutated according to the evolutionary rate of its own evolutionary chain, which is described as follows:

Let x an m-th generation individual, M and ε are an integer and a real number, respectively. If $m > M$ and the following inequality holds, then x is considered unpromising to generate good offsprings, and it is to be mutated.

$$\frac{\sum_{i=1}^{M}[f(x_{-(i-1)}) - f(x_{-i})]}{M} = \frac{f(x_0) - f(x_{-M})}{M} < \varepsilon \tag{1}$$

where $f(x_i)$ is the evaluation of x_i, x_{-i} is $x_{-(i-1)}$'s father, x_0 is x, M and ε are constants depending on the problems in hand.

The evolutionary chain of a promising individual is called a well evolving chain, and the one of an unpromising individual is called a badly evolving chain. Figures 1 and 2 give some curves of well and badly evolving chains respectively. In the above method, a promising individual's schema is preserved not to be destroyed. Contrarily, an unpromising individual is mutated to generate a new start for search.

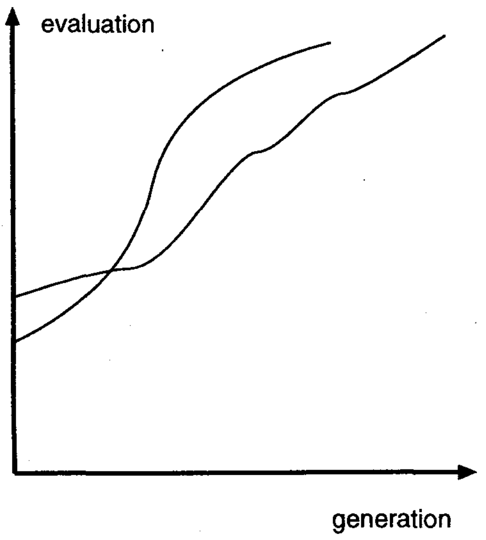


Fig.1 Well evolving chains

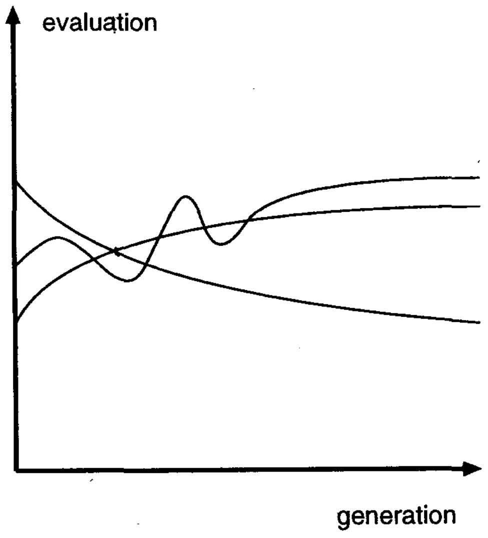


Fig.2 Badly evolving chains

3 CONTROL STRATEGY OF REASONABLE COMPETITION

Crossover and mutation are two main operations in genetic algorithm. Crossover has the property of searching in local areas. With the reiteration of crossover, some locally optimal solutions are found. Mutation provides some new starts for the search. In traditional genetic algorithm, mutated individuals in every iteration are immediately inserted into the current population. We think that too high frequency of inserting mutated individuals, as new starts, may disturb the crossover to search locally optimal solutions. On the other hand, if a mutated individual is immediately inserted into the current population, it may be soon eliminated in competition with the individuals which have evolved for many generations. We know a new and transiently weak individual may be a promising individual, and an old and transiently strong individual may not be a really promising individual.

In order to bring the effects of crossover and mutation into full play, we propose a control strategy of reasonable competition — competing only among the individuals which have the same length of evolutionary chain:

1) The mutated individuals are not immediately inserted into the current population P, but are inserted into a set S_m, so that the crossover cannot be disturbed too frequently by mutation.

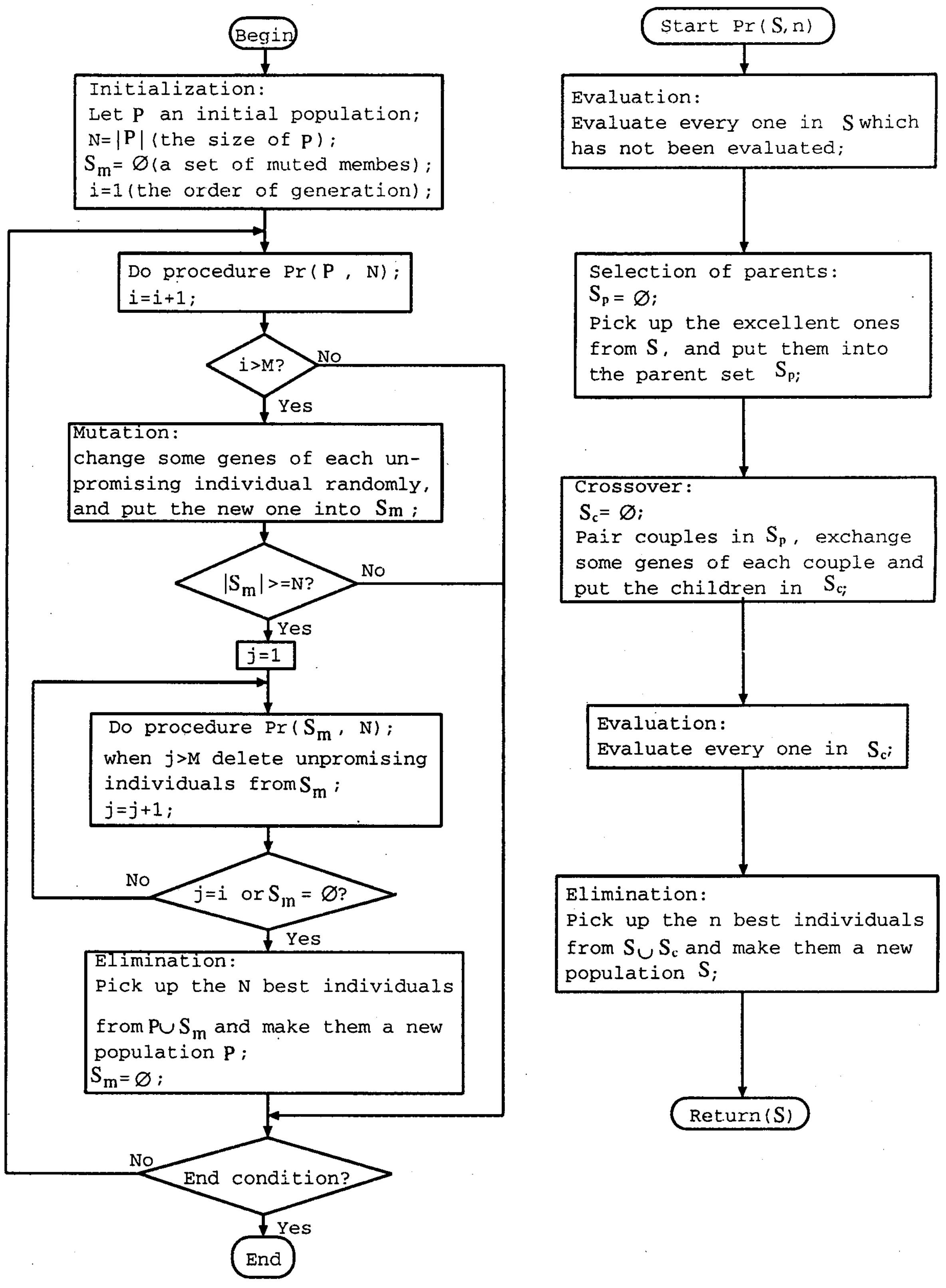
2) The competition (selection, and elimination) is independently done in P or in S_m, where the lengths of evolutionary chains of individuals in P are the same, and so are the ones in S_m.

3) The individuals in S_m are inserted in P only when they have evolved to the same generation as the ones in P and they are promising by inequality (1).

For details, see the algorithm in the next section.

4 THE GENETIC ALGORITHM WITH EVOLUTIONARY CHAIN-BASED MUTATION

According to the discussion in the last section we give the genetic algorithm with evolutionary chain-based mutation (ECMGA) as follows:


Begin
Initialization: Let P an initial population; N=|P|(the size of P); S_m= ∅(a set of muted membes); i=1(the order of generation);
Do procedure Pr(P , N); i=i+1;
i>M?
No
Yes
Mutation: change some genes of each un-promising individual randomly, and put the new one into S_m;
$|S_m|$>=N?
No
Yes
j=1
Do procedure Pr(S_m, N); when j>M delete unpromising individuals from S_m; j=j+1;
j=i or S_m = ∅?
No
Yes
Elimination: Pick up the N best individuals from P∪S_m and make them a new population P; S_m=∅;
End condition?
No
Yes
End
Start Pr(S,n)
Evaluation: Evaluate every one in S which has not been evaluated;
Selection of parents: S_p = ∅; Pick up the excellent ones from S, and put them into the parent set S_p;
Crossover: S_c= ∅; Pair couples in S_p, exchange some genes of each couple and put the children in S_c;
Evaluation: Evaluate every one in S_c;
Elimination: Pick up the n best individuals from S∪S_c and make them a new population S;
Return(S)

(a) The main flowchart of ECMGA (b) The procedure of no-mutation

Fig.3 The flowchart of ECMGA

Step 1. Let P an initial population, $S_m := \phi$ a set for inserting mutated individuals, $N := |P|$ the size of P, and $i := 1$;

Step 2. Do no-mutation evolution (just selection, crossover and elimination) in P; set $i := i + 1$; when $i > M$, mutate the unpromising individuals and insert them into S_m;

Step 3. Do step 2 until the size of S_m: $|S_m| \geq N$; set $j := 1$;

Step 4. Do no-mutation evolution in S_m; set $j := j + 1$; when $j > M$, delete the unpromising individuals from S_m;

Step 5. Do step 4 until $j = i$ or $S_m = \phi$; then pick up the N best individuals from $P \bigcup S_m$ and make them a new population P;

Step 6. set $S_m := \phi$ and go to step 2 unless the end condition is satisfied.

The flowchart of this algorithm is shown in Fig.3.

5 EXAMPLES

In this section, we illustrate the evolutionary efficiencies of ECMGA with the following two examples:
1) Nonlinear system identification with genetic algorithm and fuzzy modeling.
2) A typical combinatorial optimization problem—the traveling salesman problem (TSP).

5.1 Identification of Nonlinear Systems

The techniques of nonlinear system identification are applied in many fields in order to predict the behaviors of unknown systems given input-output data. The problem of nonlinear system identification is defined formally in the following way.

Assume that the single valued output y, of an unknown system, behaves as a function of n input values, i.e.

$$y = f(\mathbf{x}) \qquad (\mathbf{x} \in R^n) \tag{2}$$

where $\mathbf{x}=(x_1, x_2, \cdots, x_n)$ is a n-dimensional real vector and f ($\mathbf{x}$) is an unknown nonlinear function. Given K observations of these input-output data pairs (see Table 1), the system identification task is to approximate the true function f with $\hat{f}$.

Table 1 The input and output of a nonlinear system

INPUT				OUTPUT
$x_1(1)$	$x_2(1)$	$\cdots$	$x_n(1)$	$y(1)$
$x_1(2)$	$x_2(2)$	$\cdots$	$x_n(2)$	$y(2)$
		$\cdots$		$\cdots$
$x_1(K)$	$x_2(K)$	$\cdots$	$x_n(K)$	$y(K)$

Once this approximate function $\hat{f}$ has been estimated, a predicted output $\hat{y}$ can be found for any input vector $\mathbf{x} = (x_1, x_2, \cdots, x_n)$, i.e.

$$\hat{y} = \hat{f}(\mathbf{x}) \qquad (\mathbf{x} \in R^n) \tag{3}$$

This $\hat{f}$ is called the "complete form" of $\hat{f}$.

In our previous work, we proposed a method of fuzzy modeling by genetic algorithm with tree-structured individuals, called FMGA, to identify nonlinear systems[10]. The idea of this method is briefly as follows:

In order to identify a nonlinear system with the form (1), we do a fuzzy partition of the input area. For example, a partition of 3-dimensional input is shown in Fig.4. The area near the boundaries are "fuzzy area". For each fuzzy partition we can get a fuzzy model consisting of some fuzzy rules with the following form:

$$\text{if } x_1 \text{ is } A_1, x_2 \text{ is } A_2, \cdots, x_n \text{ is } A_n$$

$$\text{then } y = a_0 + a_1 x_1 + \cdots + a_n x_n \tag{4}$$

to estimate the output y of the system, where $A_i \in$ {small, large, arbitrary} is a fuzzy linguistic value ($1 \leq i \leq n$), and the rules are mixed by using the membership functions. Suppose that we are to cope with the statement "x_j is small", and the current domain for x_j is $[p, q]$. Then we give the membership function for "small" and "large" as

$$A_{small}(x_j) = \begin{cases} 1 & \text{if } x_j \le p + (q-p)/4 \\ \frac{2}{p-q}x_j + \frac{p+3q}{2(q-p)} & \text{if } p + (q-p)/4 \le x_j \le p + 3(q-p)/4 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

$$A_{large}(x_j) = \begin{cases} 0 & \text{if } x_j \le p + (q-p)/4 \\ \frac{2}{q-p}x_j - \frac{3p+q}{2(q-p)} & \text{if } p + (q-p)/4 \le x_j \le p + 3(q-p)/4 \\ 1 & \text{otherwise} \end{cases} \tag{6}$$

respectively. They are shown in Fig.5.

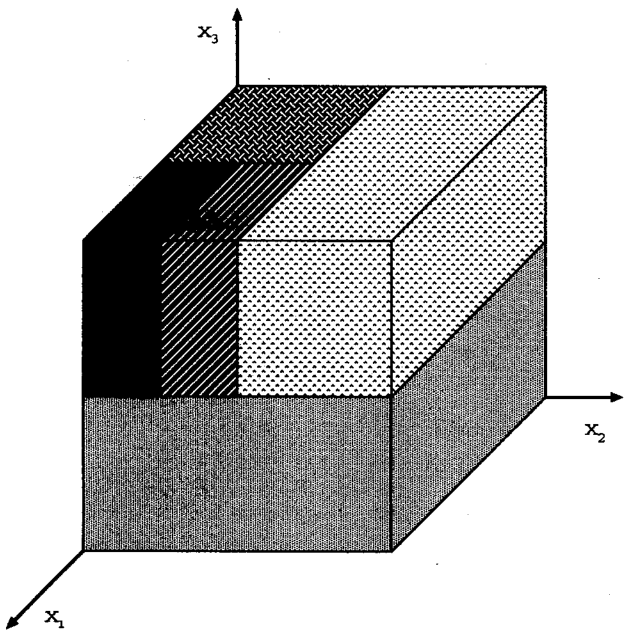


Fig.4 A fuzzy partition

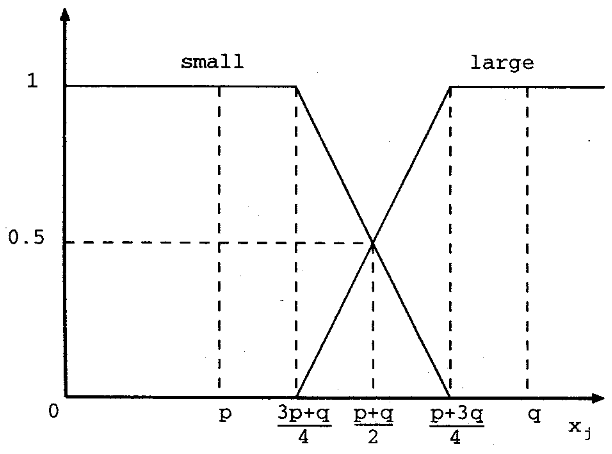


Fig.5 Fuzzy functions "*small*" and "*large*"

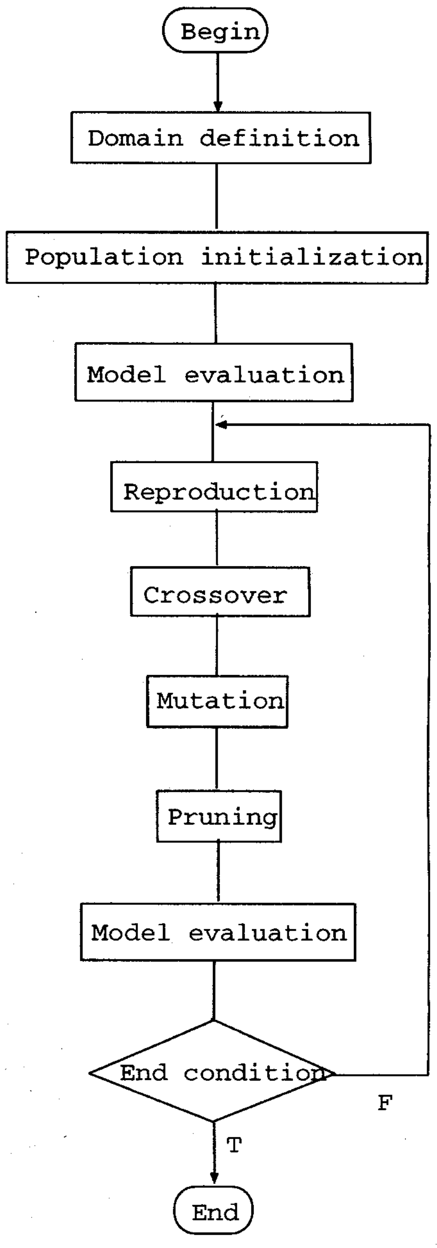


Fig.6 The flowchart of FMGA

In [10], the methods of how to get a fuzzy model from a fuzzy partition and how to get an output from a fuzzy model are provided. The mechanism of genetic algorithm is used to generate and evolve fuzzy partitions. The flowchart of the genetic algorithm FMGA used in[10] is shown in Fig.6.

Now we replace the genetic algorithm in FMGA with ECMGA described in the last section, we call this method FMECMGA, and compare the evolutionary effects of the two methods, FMGA and FMECMGA. Let us take a nonlinear function:

$$y = x_1 e^{(-x_1^2 - x_2^2)}, \qquad -2 \le x_1, x_2 \le 2. \tag{7}$$

and randomly generated 200 sets of data $\{(x_1(k), x_2(k)) | k = 1, 2, \cdots, 200\}$ in the input area. Then the evaluation function on fuzzy partition p is defined as

$$g(p) = \left(\sum_{k=1}^{200} (y(k) - \hat{y}(k,p))^2 \right)^{-1} \tag{8}$$

where $y(k)$ and $\hat{y}(k,p)$ are respectively the output values of function (7) and the fuzzy model corresponding to p.

Figure 7 shows the evaluations of best partitions in each generation by FMGA and FMECMGA. Table 2 shows the parameters and results of FMGA and FMECMGA.

Table 2 Parameters and results

	FMGA	FMECMGA
population size	50	50
p_c	0.8	0.8
p_m	0.01	
M		10
ε		0.2
generations	100	100
best evaluation	15.87	50.69

where p_c and p_m are respectively the probabilities of crossover and mutation, M and ε are the parameters in inequality (1). The initial populations of FMGA and FMECMGA were the same. 100 generations were generated both by FMGA and FMECMGA.

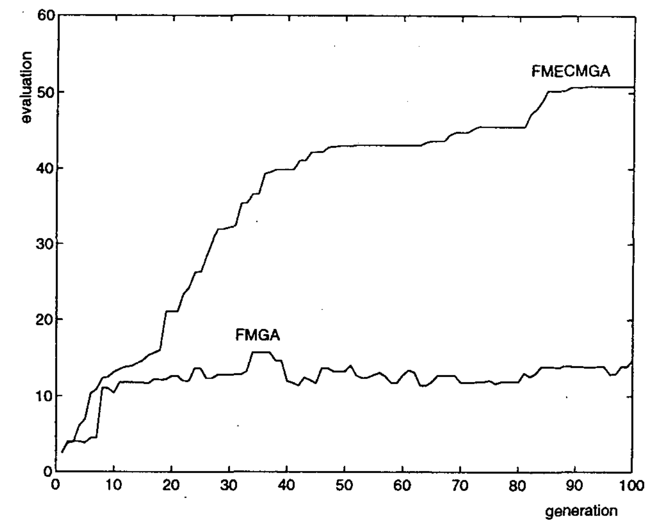


Fig.7 Comparison of FMECMGA to FMGA

The evaluation of best partition of FMGA was 15.69, which appeared at the 34th generation, and it seemed no possibility to get a better partition. The evaluation of best partition of FMECMGA was 50.87, which appeared at the 92th generation, and after the 18th generation, the evaluations of best partitions of FMECMGA were larger than 15.69. In order to compare FMECMGA with FMGA, here the "generation" of FMECMGA means the time of a no-mutation evolution having been done (see the algorithm described in the last section).

5.2 Traveling Salesman Problem

Let $G = (V, E)$ be a complete graph with weights on the edges. A Hamiltonian cycle of G is a cycle that visits each vertex of the graph exactly once. The traveling salesman problem (TSP) is the problem of finding a Hamiltonian cycle with minimum weight. A Hamiltonian cycle, for an 8-city example of the TSP is illustrated in Fig.8. The TSP is a typical NP-complete problem, easy to state, difficult to solve.

There are many studies on the encoding and crossover for the TSP (see [11]-[18]). In this paper we take the methods proposed by Yamamura et al. for encoding and crossover (see [15]). The method can be briefly expressed as follows.

1) encoding method:

A cycle is represented as the sequence of cities, i.e., the i-th alphabet represents the city which will be visited on the i-th position in the order. For example, the cycle in Fig.8 is represented as follows:

$$\overset{1}{A}\ \overset{2}{D}\ \overset{3}{B}\ \overset{4}{E}\ \overset{5}{H}\ \overset{6}{F}\ \overset{7}{G}\ \overset{8}{C}$$

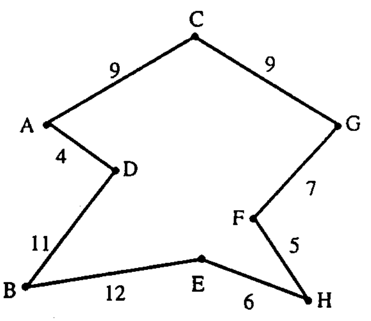


Fig.8 8-city example of TSP and a cycle

2) crossover method:
For an n-city problem, find two subpaths which contain the same cities, and lengths are greater than 1 and less than $(n-1)$ from a pair of parent cycles, then exchange these two subpaths to get two children. For example, suppose that two parent cycles P_1 and P_2 are

P_1: (A B $\underline{\text{C D E}}$ F G H) P_2: (A G $\underline{\text{D E C}}$ F B H)

We can find that two subpaths $\underline{\text{C D E}}$ and $\underline{\text{D E C}}$ of P_1 and P_2 respectively contain the same cities, exchange these two subpaths, and get two children:

C_1: (A B $\underline{\text{D E C}}$ F G H) C_2: (A G $\underline{\text{C D E}}$ F B H)

Such a pair of subpaths are referred to as a pair of inheritable genes for the pair of parents. A pair of parents may have many such genes. For example, the above pair of parents P_1 and P_2 have the inheritable genes as follows:

P_1's: (A H) (D E) (H A B) (G H A) (C D E) (B A H G) (C D E F)
(B C D E F) (C D E F G) (F G H A B) (B C D E F G) (F G H A B C)

P_2's: (A H) (D E) (B H A) (G A H) (D E C) (G A H B) (D E C F)
(D E C F B) (G D E C F) (F B H A G) (G D E C F B) (C F B H A G)

and the reverse of them are also the inheritable ones.

The methods of encoding and crossover have the following advantages as expressed in [15]:

1. the completeness of encoding;
2. the soundness of encoding;
3. the non-redundancy of encoding;
4. the character preservation of crossover.

We applied ECMGA to 100-city, 200-city, 300-city, 400-city, and 500-city problems. The cities were generated in the area $0 \le x, y \le 1$ uniformly at random. The sizes of populations were 100, 100, 100, 150, and 200 respectively. The parameters M and ε in inequality (1) were $M = 10, \varepsilon = (maxdistance - mindistance)/2M$, where $maxdistance = max\{distance(i,j)|1 \le i,j \le n\}$, $mindistance = min\{distance(i,j)|1 \le i,j \le n, i \ne j\}$, i and j represented the cities, and n was the number of cities. The best cycles for these problems are shown in Figs 9–13. They were obtained at the 894-th, 2154-th, 11050-th, 7585-th, and 6946-th loops respectively, and the distances of these cycles were 8.27, 11.37, 14.67, 16.79, and 19.09 respectively.

CGA (see section 1) seems no ability to solve the problem when the number of cities is large. Figure 14 gives the best result of CGA for the 100-city problem. It was got at the 2084-th loop, and the shortest distance is 31.9144. The probabilities of crossover and mutation in CGA here were 0.8 and 0.001 respectively. The algorithm was run for 2584 loops in total. Figure 15 is the comparison of ECMGA to CGA on the evolution efficiency of the 100-city problem.

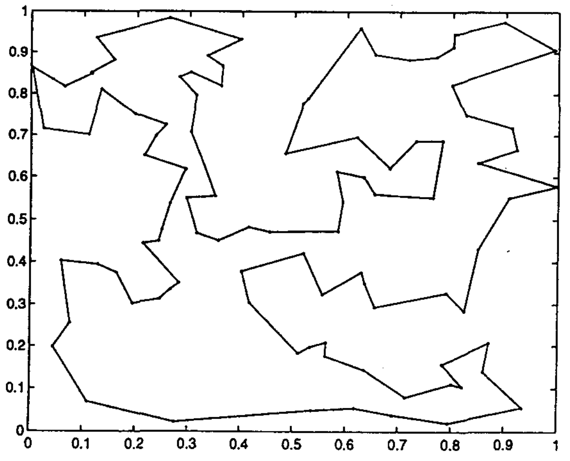


Fig.9 The best cycle by ECMGA for the TSP with 100 cities

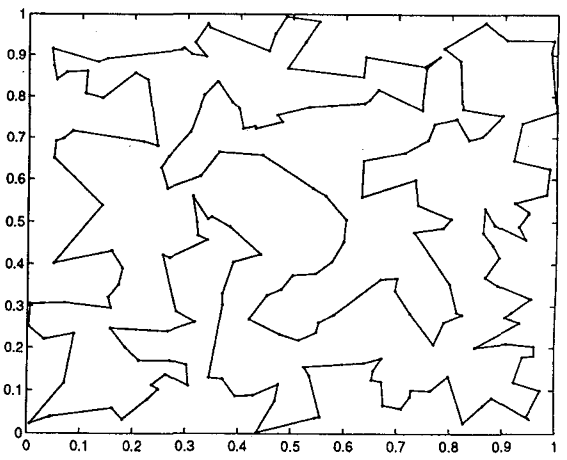


Fig.10 The best cycle by ECMGA for the TSP with 200 cities

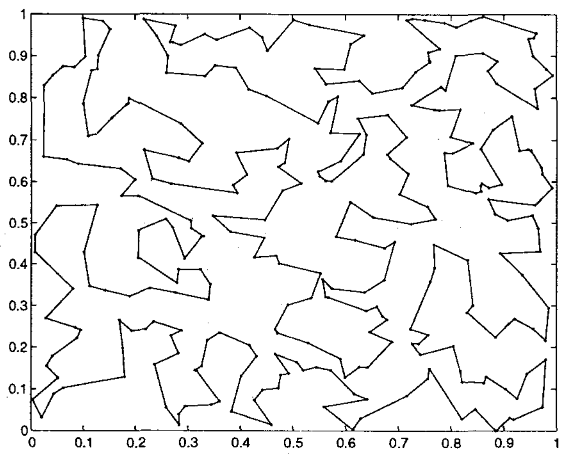


Fig.11 The best cycle by ECMGA for the TSP with 300 cities

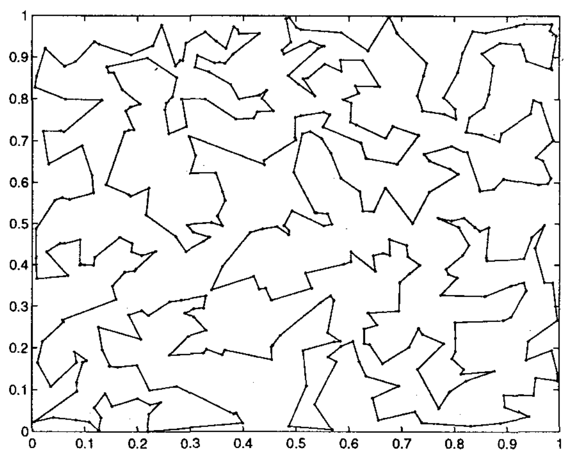


Fig.12 The best cycle by ECMGA for the TSP with 400 cities

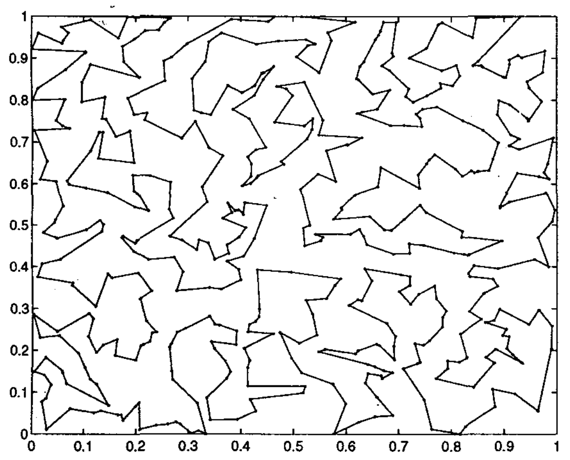


Fig.13 The best cycle by ECMGA for the TSP with 500 cities

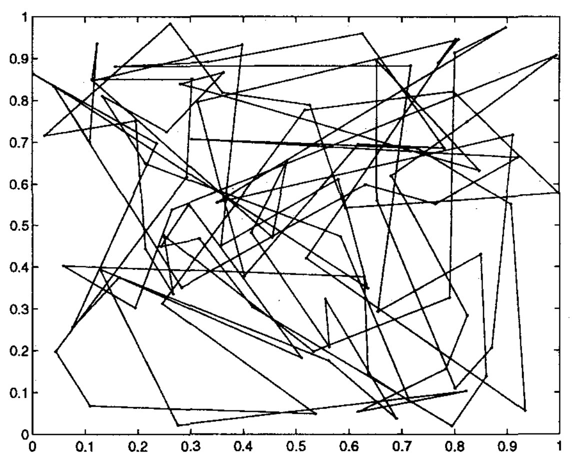


Fig.14 The best cycle CGA for the TSP with 100 cities

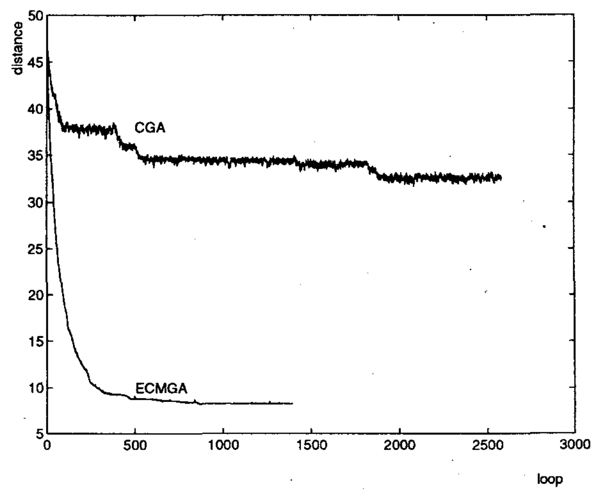


Fig.15 The evolutions of ECMGA and CGA for the TSP with 100 cities

6 CONCLUSION

The genetic algorithm with evolutionary chain-based mutation is proposed, in which the importance of heuristic information provided by the evaluation function is emphasized. In this way the blindness of stochastic mutation is avoided. It becomes possible that the evolutionary process can be controlled under human's intelligence, knowledge and strategy, such as the control rules like inequality (1) in section 2. Two examples, nonlinear system identification problem and the traveling salesman problem, are given to show the good performance of this method.

REFERENCES

[1] J. J. Grefenstette (ed.): "Proc. of the First Int. Conf. on Genetic Algorithms and Their Applications", Lawrence Erlbaum associates, 1985.

[2] J. J. Grefenstette (ed.): "Proc. of the Second Int. Conf. on Genetic Algorithms", Lawrence Erlbaum associates, 1987.

[3] J. D. Schaffer (ed.): "Proc. of the Third Int. Conf. on Genetic Algorithms", Morgan Kaufmann Publishers, 1989.

[4] R. K. Belew (ed.): "Proc. of the Fourth Int. Conf. on Genetic Algorithms", Morgan Kaufmann Publishers, 1991.

[5] S. Forrest (ed.): "Proc. of the Fifth Int. Conf. on Genetic Algorithms", Morgan Kaufmann Publishers, 1993.

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

[7] K. A. De Jong: "Foundations of Genetic Algorithms·2", Morgan Kaufmann Publishers (1993), 5.

[8] M. Srinivas, and L. M. Patnaik: IEEE Computer Magazine, **27** (1994), 6, 17.

[9] M. D. Vose: Annals of Mathematics and Artificial Intelligence, **10** (1994), 4, 423.

[10] M. Tanaka, J. Ye and T. Tanino: "Fuzzy modelling by genetic algorithm with tree-structured individuals," accepted for publication in International Journal of Systems Science.

[11] I. M. Oliver, D. J. Smith and J. R. C. Holland: "Proc. of the Second Int. Conf. on Genetic Algorithms, Cambridge, 1987", ed. J. J. Grefenstette, Lawrence Erlbaum associates (1987), 224.

[12] J. Y. Suh and D. V. Gucht: "Proc. of the Second Int. Conf. on Genetic Algorithms, Cambridge, 1987", ed. J. J. Grefenstette, Lawrence Erlbaum associates (1987), 100.

[13] P. Jog, J. Y. Suh and D. V. Gucht: "Proc. of the Third Int. Conf. on Genetic Algorithms, Fairfax, 1989", ed. J. D. Schaffer, Morgan Kaufmann Publishers (1989), 110.

[14] D. Whitley, T. Starkweather and D. Fuquay: "Proc. of the Third Int. Conf. on Genetic Algorithms, Fairfax, 1989", ed. J. D. Schaffer, Morgan Kaufmann Publishers (1989), 133.

[15] M. Yamamura, T. Ono and S. Kobayashi: Journal of the Japanese Society of Artificial Intelligence, **7** (1992), 6, 117 (in Japanese).

[16] A. Homaifar, S. Guan and G. Liepins: "Proc. of the Fifth Int. Conf. on Genetic Algorithms, Springfield, 1993", ed. S. Forrest, Morgan Kaufmann Publishers (1993), 460.

[17] T. N. Bui and B. R. Moon: "Proc. of the First IEEE Conf. on Evolutionary Computation, Orlando, 1994", ed. J. D. Schaffer, IEEE Service Center (1994), 7.

[18] H. Tamaki, H. Kita, N. Shimizu, K. Maekawa and Y. Nishikawa: "Proc. of the First IEEE Conf. on Evolutionary Computation, Orlando, 1994", ed. J. D. Schaffer, IEEE Service Center (1994), 1.