Applied Soft Computing 43 (2016) 415–423

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Multi-offspring genetic algorithm and its application to the traveling salesman problem☆

Jiquan Wang[a,b,*], Okan K. Ersoy[a], Mengying He[c], Fulin Wang[c]

[a] Parallel Distributed Processing Laboratory, School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907-1285, USA
[b] The College of Engineering, Northeast Agricultural University, Harbin 150030, China
[c] Management Science and Engineering Complex Laboratory, College of Engineering, Northeast Agricultural University, Harbin 150030, China

ARTICLE INFO

Article history:
Received 28 September 2015
Received in revised form
28 December 2015
Accepted 15 February 2016
Available online 2 March 2016

Keywords:
Multi-offspring
Genetic Algorithm
TSP
Crossovers
Mutations

ABSTRACT

The paper provides a multi-offspring genetic algorithm (MO-GA) in accordance with biological evolutionary and mathematical ecological theory, and illustrates its application in the traveling salesman problem (TSP) in comparison to the basic genetic algorithm (BGA). In MO-GA, the number of offsprings is significantly increased as compared to the BGA. MO-GA increases the probability of producing excellent individuals, and also makes the population more competitive, thus yielding considerable improvement. Test results with six TSP examples show that MO-GA has faster speed, and the number and time of iterations are significantly reduced as compared to the BGA.

1. Introduction

The traveling salesman problem (TSP) is a well-known and important combinatorial optimization problem [1]. The goal is to find the shortest tour that visits each city in a given list exactly once and then returns to the starting city. This is a typical NP-hard problem [2–4], which has extremely large search spaces and is very difficult to solve. Apart from its theoretical approach, TSP is widely used as a model in many fields such as vehicle routing [5], scheduling problems [6], design of integrated circuits [7], physical mapping problems [8], constructing phylogenetic trees [9], machine flow shop scheduling [10] and so on. Hence, solving TSP has significant practical implications. There have been many studies of intelligent algorithms for TSP since K. Menger first presented it in 1932 [11]. In 1985, Goldberg and Grefenstette first used the genetic algorithm for solving the TSP problem, and put forward three crossover methods [12,13].

TSP problems can be classified into two categories. One category involves distances between cities which are symmetrical. In other words, supposing the distance between city i and city j is d_{ij}, then, we have $d_{ij} = d_{ji}$. The other category involves distance between two cities which are non-symmetrical, i.e. $d_{ij} \neq d_{ji}$. In this paper, we discuss the symmetrical TSP problem. A good reference for a detailed discussion of TSP is by Lawer et al. [14]. An excellent review article on GA for TSP is by Larranaga et al. [15].

In recent years, many scholars came up with some improved GA algorithms. In 1991, Whitley D., Starkweather T., and Shaner D. proposed a new solution for TSP using GA in their "The Traveling Salesman and Sequence Scheduling: Quality Solutions Using Genetic Edge Recombination". In 2003, Lalena M. similarly solved the TSP using GA. In 2005, Takahashi R. solved the TSP through genetic algorithms with changing crossover operators. In 2007, Ding chao, Cheng ye and He Miao presented a paper for solving another type of TSP called the clustered traveling salesman problem. In 2009, Vahdati Go., Yaghoubi M. Poostchi M., Naghibi M.B. presented a GA with heuristic crossover and mutation operator. In 2009, Jinqiu Yang, Jiangang Yang, Genlang Chen presented an adaptive clustering method and a novel genetic algorithm for TSP. In 2011, Kaya Y., Uyar M. and Tekin R. presented a ring crossover operator. In 2011, Deep K., Mebrahtu H. presented combined mutation operators of GA for TSP. In 2013, Chunhui Zhou, Shijun Hu, Yuanqiao introduced the idea of simulated annealing to improve the mutation operator of GA. In 2014, Rani K., Kumar V. presented

☆ This work was supported in part by sub project of National Science and Technology support program under Grant 2014BAD06B01-23, and in part by the Northeast Agricultural University Graduate Science and Technology Innovation Project under Grant yjscx14024.

* Corresponding author at: The College of Engineering, Northeast Agricultural University, Harbin 150030, China. Tel.: +86 45155191737.
E-mail addresses: wang-jiquan@163.com (J. Wang), ersoy@purdue.edu (O.K. Ersoy), 857268690@qq.com (M. He), fulinwang1462@126.com (F. Wang).

http://dx.doi.org/10.1016/j.asoc.2016.02.021
1568-4946/

a new crossover operator which is variation of order crossover (OX). These improved GA methods overcame premature convergence and stagnation in a local optimum to some extent.

In the literature for genetic algorithms, the number of generated offsprings is the same as the number of parents. For the survival and diversity of the species, it should be desirable to generate more number of offsprings. Aiming at this problem, the paper provides a multi-offspring genetic algorithm (MO-GA), and discusses how this can be done. The simulation results with test functions show that MO-GA indeed generates improved results as compared to the basic genetic algorithm in the TSP application.

2. Theoretical foundation of MO-GA

2.1. Biological theory foundation

In the literature, the genetic algorithm for TSP is discussed in terms of a pair of parents generating a pair of offsprings [16–19]. However, in the process of biological evolution, the number of offsprings restricted to be equal to or less than a pair of parents is not common. Even though this is possible for reasons such as disease, food, water, and other factors, the species may become extinct or be restricted in terms of genetic competition. When the number of offsprings is larger than 2, the population size grows larger and become more competitive, with increased chances of better offsprings. Thus, it can be desirable to have the number of offsprings larger than 2 in developing the genetic algorithm.

2.2. Mathematical ecological theory foundation

In order to illustrate the probability of species extinction, suppose one species has only one individual at first. Then at a certain time t, the probability of population size equaling 0 is given by

$$p_0(t|i=1)=\frac{\mu e^{(\lambda-\mu)t}-\mu}{\lambda e^{(\lambda-\mu)t}-\mu} \tag{1}$$

where i is the size of the initial population, μ is the mortality rate, and λ is the reproduction rate.

The probability for the population whose initial size equals i to be extinct with time elapsing is given by

$$p_0(t)=[p_0(t|i=1)]^i=\left[\frac{\mu e^{(\lambda-\mu)t}-\mu}{\lambda e^{(\lambda-\mu)t}-\mu}\right]^i \tag{2}$$

As t tends to infinity, there are three situations as follows:

(1) When the reproduction rate is less than the mortality rate, that is to say $\lambda<\mu$, the exponential term in Eq. (2) would tend to 0 as $t\to\infty$, resulting in

$$\lim_{t\to\infty}(p_0(t))=1 \tag{3}$$

Hence, the species must become extinct as time grows.

(2) When the reproduction rate is greater than the mortality rate, that is to say $\lambda>\mu$, Eq. (2) as t tends to ∞ can be represented as follows:

$$p_0(t)\to\left[\frac{\mu e^{(\lambda-\mu)t}}{\lambda e^{(\lambda-\mu)t}}\right]^i=\left[\frac{\mu}{\lambda}\right]^i \tag{4}$$

According to Eq. (4), such population cannot guarantee to continue to exist because there is still a finite probability of extinction. However, if the reproduction rate is much greater than the mortality rate, and if the initial population size is larger, the biological extinction probability will be smaller.

(3) When the reproduction rate equals the mortality rate, that is to say, $\lambda=\mu$, Eq. (2) can be expanded in a series of exponential terms. Letting $\lambda-\mu=r$, $p_0(t)$ as t tends to ∞ can be written as

$$p_0(t)=\left[\frac{\mu(rt+r^2t^2/2!+\cdots)}{\lambda(1+rt+r^2t^2/2!+\cdots)-\mu}\right]^i \tag{5}$$

When $r\to 0$, meaning $\mu\to\lambda$, and ignoring r^2, we get

$$p_0(t)\to\left[\frac{\mu rt}{(\lambda-\mu)+\lambda rt}\right]^i\to\left(\frac{\lambda t}{1+\lambda t}\right)^i \tag{6}$$

Hence,

$$\lim_{t\to\infty}\left(\frac{\lambda t}{1+\lambda t}\right)^i=1 \tag{7}$$

Even when the reproduction rate equals the mortality rate, Eq. (7) shows that the species must become extinct. Although the population expected size is constant, while randomly fluctuating around the population expected size, the species will be extinct long after. Only when $\lambda>\mu$, the population may survive forever. Thus, the probability distribution of biological population size depends on the product of reproduction rate and time when the biological initial population is known. Therefore, in order to get more excellent individuals in possible shortest time, there is a need to improve the reproduction rate of species.

According to mathematical ecological theory in biology, if the number of parents is the same as the number of offsprings for a certain population, the probability of population extinction is 1. Since BGA artificially controls population size that is unchangeable, the population of BGA does not have possibility of extinction. Hence, the evolutionary law of BGA is not consistent with the evolutionary law of natural biology. In contrast, the evolutionary law of MO-GA is consistent with the evolutionary law of natural biology, and thus MO-GA has sufficient biological theory foundation.

3. Multi-offspring genetic algorithm

Based on biological theory foundation and mathematical ecological theory foundation, MO-GA is proposed to solve the TSP. The difference between MO-GA and BGA has to do with the way crossover and mutation operations are carried out. In the existing literature for BGA, the crossover operation is done with two parents generating two offsprings, whereas two parents generate more than two offsprings in MO-GA. In the mutation operation of MO-GA, $2n$ offsprings are mutated in accordance with a certain mutation probability. In order to maintain the population size unchangeable, n outstanding individuals are selected and kept at the end of each generation.

3.1. Evolutionary strategy of MO-GA

The initial computations of MO-GA for TSP are as follows:

(a) The parameters of MO-GA are initiated, such as the mutation rate equal to 0.3, and the initial population equal to n. The number of cities m is chosen according to the current application.

(b) If the distance between any two cities is given directly, the route distance can be computed. Otherwise, the total distance of each path for TSP of m cities is calculated according to the coordinates of the cities as follows: if the coordinates of any two cities i and k are (x_i,y_i) and (x_k,y_k), the distance between them is $D_{ik}=[(x_i-x_k)^2+(y_i-y_k)^2]^{0.5}$.

For example, if $m = 9$, and the route is 691387452, the total distance of this path is

$$D_{69} + D_{91} + D_{13} + D_{38} + D_{87} + D_{74} + D_{52} + D_{26}$$

The succeeding evolutionary strategy of MO-GA is as follows:

(1) Generate the initial population of size n. Sort the members in ascending order according to their path lengths.
(2) Select q elitist members with best ranking among n parents.
(3) Using the crossover operator, generate $2n$ offsprings. This is discussed in Section 3.3.
(4) Generate a new population consisting of the $2n$ offsprings and the q elitist members chosen in Step 2.
(5) Sort the members of the new population in ascending order according to their path lengths.
(6) Select q elitist members with best ranking in the new population.
(7) Using the mutation operator, modify the $2n$ offsprings generated by crossover. This is discussed in Section 3.4.
(8) Regenerate the new population consisting of the $2np_m$ mutated offsprings, $2n(1-p_m)$ unmutated offsprings and q elitist members chosen in Step 6.
(9) Sort the members of the regenerated new population in ascending order according to their path lengths.
(10) If the stop condition is met, output the best route, the total distance of m cities and terminate the loop; otherwise
(11) Select n members with best ranking within the last population. Go to step 2 to start the next loop.

The evolutionary strategy flow diagram for MO-GA is shown in Fig. 1.

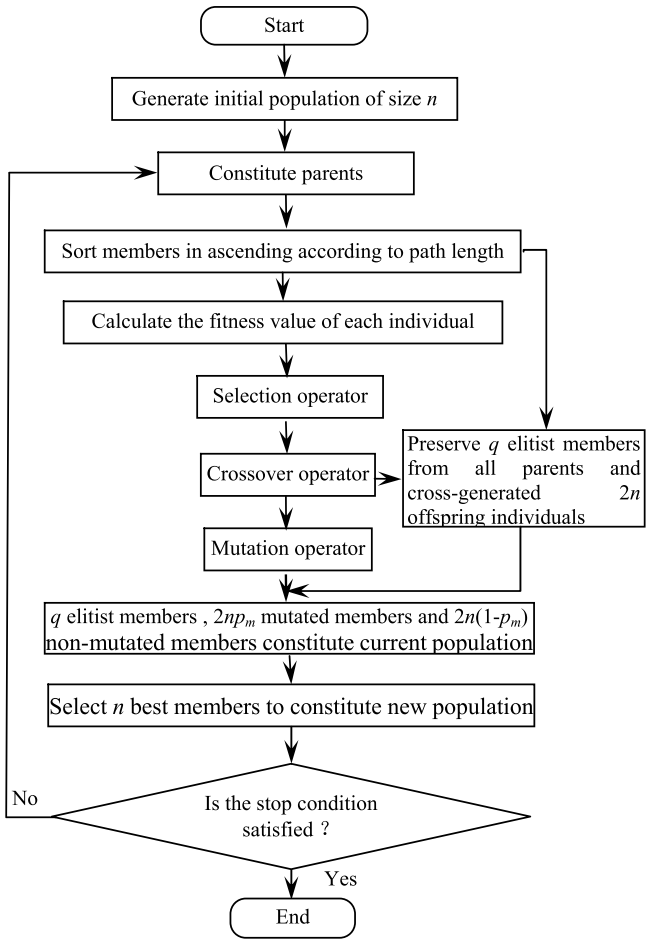


Fig. 1. The flow diagram of MO-GA.

3.2. Initialization of population

We will further discuss MO-GA in terms of the TSP. The traveling route can be represented by the order of the cities, and the cities can be indicated by an array of integers. For example, we will assume there are nine cities in a TSP. We can represent a route by random non-repeating integers between 1 and 9, such as 3 2 5 9 6 1 7 8 4. More generally, if there are m cities, and the chosen population size is n, we initially generate n randomly chosen routes each of which contains m non-repeating integers.

3.3. Crossover operations

The roulette wheel method was used to pair members [20]. For this purpose, the fitness function value and the corresponding probability per member is calculated.

There are two models used for the computation of the fitness value. The first model with a population of size n is given by

$$F_1(X_i') = \beta(1-\beta)^{i-1}, \quad i = 1, 2, \ldots, n \tag{8}$$

where X_i' is the ith member of the population sorted according to the fitness values in ascending order, and $\beta \in (0, 1)$ is a parameter usually chosen between 0.01 and 0.3 [21].

The second model is given by

$$F_2(X_i') = \frac{n-i+1}{n}, \quad i = 1, 2, \ldots, n \tag{9}$$

We use the roulette wheel method to select to crossover members as follows:

Suppose the fitness value is $F(X_i')$. Then, the selection probability of the ith member is given by

$$P_i = \frac{F(X_i')}{\sum_{i=1}^{n} F(X_i')} \tag{10}$$

Letting

$$PP_0 = 0 \tag{11}$$

$$PP_i = \sum_{j=1}^{i} P_i, \quad i = 1, 2, \ldots, n \tag{12}$$

The roulette wheel is rotated up to n times, and a random number $\eta_k \in (0, 1)$ is generated at each rotation. When this random number satisfies

$$PP_{i-1} \le \eta_k < PP_i \tag{13}$$

The ith member is selected to take part in crossover.

When the roulette wheel method is used to pair members, the fitness function is used. If the fitness values are calculated by Eq. (8), the highly ranked members in the population are first chosen to participate in crossover since they have greater probability, and the latter members in the population are chosen less frequently since they have lower probability. This process tends to result in fast convergence, On the other hand, as iterations grow from generation to generation, the diversity of the population tends to become limited, resulting in convergence to a local minimum. If the fitness values

of members are calculated according to Eq. (9), the crossover probability of members in the population is the same, and the diversity of the population is rich, but the convergence speed is slow.

In order to achieve both fast convergence and rich population diversity as well as to avoid convergence to a local minimum, the following method was adopted: when the iteration number is odd, the fitness values are calculated by Eq. (8); When the iteration number is even, the fitness values are calculated by Eq. (9).

The procedure for generating $2n$ offsprings from a population of size n is discussed below with an example based on a 9-city TSP. Two members A and B are chosen from the population to become parents as discussed above. Suppose they are represented as

$A = 3\ \ 4\ \ 6\ \ 8\ \ 1\ \ 2\ \ 9\ \ 7\ \ 5$
$B = 2\ \ 7\ \ 1\ \ 9\ \ 5\ \ 3\ \ 6\ \ 4\ \ 8$

Two intersection points are chosen by using a uniform random number generator. Representing them as "|", an example would be

$A = 3\ \ 4\ \ 6\,|\,8\ \ 1\ \ 2\ \ 9\,|\,7\ \ 5$
$B = 2\ \ 7\ \ 1\,|\,9\ \ 5\ \ 3\ \ 6\,|\,4\ \ 8$

Next, crossover operations are utilized in two stages to generate four offsprings. In stage 1, the crossover operations are as follows:

(1) Leave the middle sections between the two intersection points intact to get

$C_1 = *\ \ *\ \ *\,|\,8\ \ 1\ \ 2\ \ 9\,|\,*\ \ *$
$C_2 = *\ \ *\ \ *\,|\,9\ \ 5\ \ 3\ \ 6\,|\,*\ \ *$

(2) Rotate the third section to the beginning of the route in each member. For example, B becomes 4-8-2-7-1-9-5-3-6.
(3) Delete the elements in B which are the same as the elements of the middle section in A. Repeat this process by exchanging A and B. For example, the middle section of A is 8-1-2-9. Removing these elements from B gives 4-7-5-3-6. Similarly, the middle section of B is 9-5-3-6. Removing these elements from A gives 7-4-8-1-2.
(4) Let k be the number of elements in the last section. Move the first k elements of the first member obtained in step 3 to become the new last section of the second member, and vice versa. The initial middle sections are exchanged to become the new middle sections. The remaining elements of the first section are also exchanged between the two members.

Using this procedure, the resulting offsprings become

$C_1 = 8\ \ 1\ \ 2\,|\,9\ \ 5\ \ 3\ \ 6\,|\,7\ \ 4$

and

$C_2 = 5\ \ 3\ \ 6\,|\,8\ \ 1\ \ 2\ \ 9\,|\,4\ \ 7$

The crossover operations discussed above are the same as the crossover operations used in the basic genetic algorithm.

In stage 2, the crossover operations are as follows:

(1) Exchange sections 1 and 2. This yields

AA = 8-1-2-9-3-4-6-7-5

BB = 9-5-3-6-2-7-1-4-8

(2) Find section 3 elements in AA, delete the corresponding ones in BB, and vice versa. Make section 3 elements in AA the section 3 elements of BB, and vice versa.

Using this procedure, the resulting offsprings become

$C_3 = 1\ \ 2\ \ 9\,|\,3\ \ 6\ \ 7\ \ 5\,|\,4\ \ 8$

$C_4 = 9\ \ 3\ \ 6\,|\,2\ \ 1\ \ 4\ \ 8\,|\,7\ \ 5$

In this way, two parents generate four offsprings by crossover operations. This means $2n$ new offsprings are generated from an initial population of size n.

3.4. Mutation operations

After the crossover operations, the $2n$ offsprings are modified with the mutation operator according to the mutation probability value of p_m. This results in $2np_m$ offsprings modified by the mutation operator.

For each member in the population, we generate a uniform random number $\doteq$. If $\delta \leq p_m$, the member is mutated. Otherwise, the member is not mutated. Thus, we can get $2np_m$ offsprings modified by the mutation operator.

The mutation operation is as follows:

Suppose DD is chosen from the population to participate in the mutation operation, and it has the representation

$DD = 6\ \ 5\ \ 8\ \ 1\ \ 4\ \ 3\ \ 9\ \ 2\ \ 7$

Two mutation points are chosen by using a uniform random number generator. Representing them as "|", an example would be

$DD = 6\ \ 5\,|\,8\ \ 1\ \ 4\ \ 3\,|\,9\ \ 2\ \ 7$

The middle section is reversed, resulting in

$EE = 6\ \ 5\,|\,3\ \ 4\ \ 1\ \ 8\,|\,9\ \ 2\ \ 7$

$2np_m$ offsprings are mutated in this way. The $2np_m$ members plus $2n(1-p_m)$ members without mutation and q elitist individuals retained after crossover make up the new population of size $2n+q$.

4. Basic genetic algorithm

MO-GA is based on ordered crossover. If we also select BGA based on ordered crossover, MO-GA and BGA have comparability. BGA is described as follows:

4.1. Evolutionary strategy of BGA

The initial computations of BGA for TSP are as follows:

(a) The parameters of BGA are initiated such as the mutation rate equal to 0.3, and the initial population size equal to n. The number of cities m is chosen according to the current application.
(b) If the distance between any two cities is given directly, the route distance can be computed. Otherwise, the computation method is the same as in Section 3.1.

The succeeding evolutionary strategy of BGA is as follows:

(1) Generate the initial population of size n. Sort the members in ascending order according to their path lengths.
(2) Select q elitist members with best ranking among n parents.
(3) Using the crossover operator, generate n offsprings. This is discussed in Section 4.3.
(4) Generate a new population consisting of the n offsprings and the q elitist members chosen in Step (2).
(5) Sort the members of the new population in ascending order according to their path lengths.

(6) Select q elitist members with best ranking in the new population.
(7) Using the mutation operator, modify the n offsprings generated by crossover. This is discussed in Section 4.4.
(8) Regenerate the new population consisting of the np_m mutated offsprings, $n(1-p_m)$ unmutated offsprings and q elitist members chosen in Step (6).
(9) Sort the members of the regenerated new population in ascending order according to their path lengths.
(10) If the stop condition is met, output the best route, the total distance of m cities and terminate the loop; otherwise
(11) Select n members with best ranking within the last population. Go to step (2) to start the next loop.

The evolutionary strategy flow diagram for BGA is shown in Fig. 2.

4.2. Initialization of population

BGA and MO-GA are the same in the population initialization method.

4.3. Crossover operations

We use the roulette wheel method to select to crossover members, as discussed in Section 3.3. The fitness value is calculated according to formula (8).

The procedure for generating n offsprings from a population of size n is discussed below with an example based on a 9-city TSP.

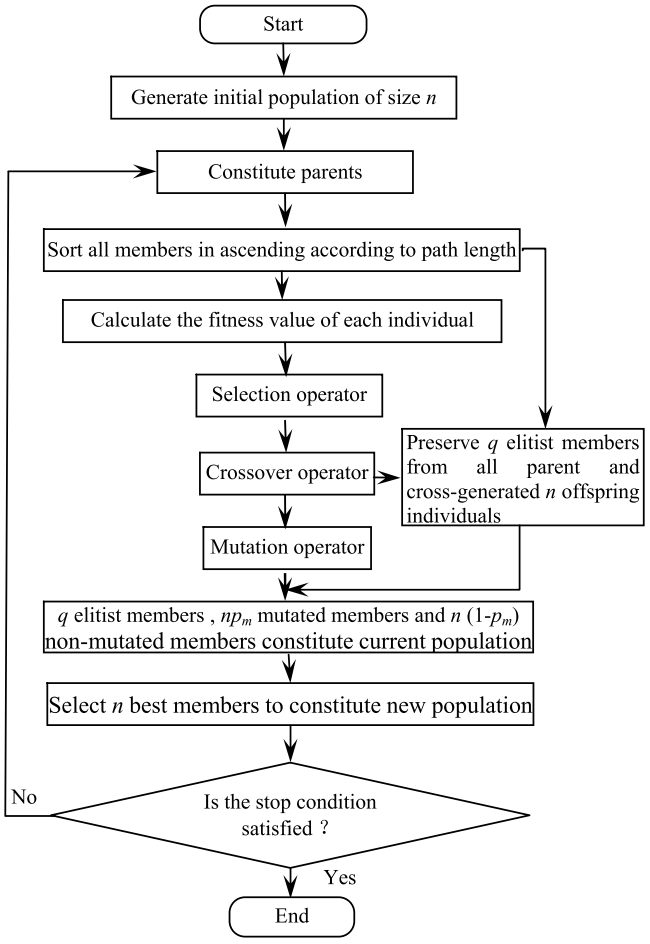


Fig. 2. The flow diagram of BGA.

Two members A and B are chosen from the population to become parents as discussed above. Suppose they are represented as

$A = 3\ \ 4\ \ 6\ \ 8\ \ 1\ \ 2\ \ 9\ \ 7\ \ 5$
$B = 2\ \ 7\ \ 1\ \ 9\ \ 5\ \ 3\ \ 6\ \ 4\ \ 8$

Two intersection points are chosen by using a uniform random number generator. Representing them as "|", an example would be

$A = 3\ \ 4\ \ 6\,|\,8\ \ 1\ \ 2\ \ 9\,|\,7\ \ 5$
$B = 2\ \ 7\ \ 1\,|\,9\ \ 5\ \ 3\ \ 6\,|\,4\ \ 8$

Next, crossover operations are utilized in two stages to generate four offsprings. In stage 1, the crossover operations are as follows:

(1) Leave the middle sections between the two intersection points intact to get $C_1 = *\ \ *\ \ *\,|\,8\ \ 1\ \ 2\ \ 9\,|\,*\ \ *$, $C_2 = *\ \ *\ \ *\,|\,9\ \ 5\ \ 3\ \ 6\,|\,*\ \ *$
(2) Rotate the third section to the beginning of the route in each member. For example, B becomes 4-8-2-7-1-9-5-3-6.
(3) Delete the elements in B which are the same as the elements of the middle section in A. Repeat this process by exchanging A and B. For example, the middle section of A is 8-1-2-9. Removing these elements from B gives 4-7-5-3-6. Similarly, the middle section of B is 9-5-3-6. Removing these elements from A gives 7-4-8-1-2.
(4) Let k be the number of elements in the last section. Move the first k elements of the first member obtained in step 3 to become the new last section of the second member, and vice versa. The initial middle sections are exchanged to become the new middle sections. The remaining elements of the first section are also exchanged between the two members.

Using this procedure, the resulting offsprings become

$C_1 = 8\ \ 1\ \ 2\,|\,9\ \ 5\ \ 3\ \ 6\,|\,7\ \ 4$

And

$C_2 = 5\ \ 3\ \ 6\,|\,8\ \ 1\ \ 2\ \ 9\,|\,4\ \ 7$

The crossover operations discussed above are the same as the crossover operations used in the basic genetic algorithm.

4.4. Mutation operations

After the crossover operations, the n offsprings are modified with the mutation operator according to the mutation probability value of p_m. This results in np_m offsprings modified by the mutation operator.

For each member in the population, we generate a uniform random number β. If $\beta \le p_m$, the member is mutated. Otherwise, the member is not mutated. Thus, we can get np_m offsprings modified by the mutation operator.

The mutation operation is as follows:

Suppose DD is chosen from the population to participate in the mutation operation, and it has the representation

$DD = 6\ \ 5\ \ 8\ \ 1\ \ 4\ \ 3\ \ 9\ \ 2\ \ 7$

Two mutation points are chosen by using a uniform random number generator. Representing them as "|", an example would be

$DD = 6\ \ 5\,|\,8\ \ 1\ \ 4\ \ 3\,|\,9\ \ 2\ \ 7$

The middle section is reversed, resulting in

$EE = 6\ \ 5\,|\,3\ \ 4\ \ 1\ \ 8\,|\,9\ \ 2\ \ 7$

np_m offsprings are mutated in this way. The np_m members plus $n(1-p_m)$ members without mutation and q elitist individuals retained after crossover make up the new population of size $n+q$.

5. Performance analysis of MO-GA and BGA

5.1. Performance analysis based on schema theorem

In MO-GA, the number of cross-generated offsprings is usually an integer multiple of the number of the parents. Letting n be the number of parents, n_1 be the number of cross-generated offsprings in MO-GA, we have

$$n_1 = \alpha n \quad \alpha \in \{2, 3, 4, \ldots\} \tag{14}$$

where α is a proportionality constant.

Definition 1. A schema S is a template that identifies a subset of strings (chromosomes) with similarities at certain string positions.

In strings with binary code, a schema is a character string based on a character set which include three characters (0, 1, *) in which the symbol * represents an arbitrary character (0 or 1). For example, the model {*1*} describes a subset with four elements {010,011,110,111}.

Definition 2. The number of determinate positions in schema S is called schema order, denoted as $K(S)$. For example, $K(011^{**}1^{**}) = 4$.

Definition 3. The distance between the first and last determinate positions in schema S is called as defining length of schema, denoted as $l(S)$. For example, $l(011^{**}1^{**}) = 5$.

Supposing there is a population of chromosomes (strings) of size n. The population at time t is represented by $A(t)$. The number of chromosomes which include certain schema S in $A(t)$ at time t is represented by $E = E(S, t)$. In selection process, each chromosome is selected according to its fitness. Representing an arbitrary chromosome in $A(t)$ by A_i, the probability that the string A_i is selected is given by

$$p_i = \frac{f_i}{\sum_{j=1}^{n} f_i} \tag{15}$$

where f_i is the fitness value of A_i.

So we can describe the number of strings that include schema S at time $t+1$ as

$$E(S, t+1) = E(S, t) \cdot n \cdot \frac{f(S)}{\sum_{j=1}^{n} f_j} \tag{16}$$

where $f(S)$ is the average fitness value of string containing schema S at time t.

The average fitness of the population is given by

$$\bar{f} = \frac{1}{n}\sum_{j=1}^{n} f_j \tag{17}$$

Beginning with $t = 0$, suppose the fitness value of chromosome containing schema S is invariably higher than the average fitness value of the population. Letting

$$f(S) - \bar{f} = c\bar{f} \tag{18}$$

where c is a constant, the selection growth equation of schema is given by

$$E(S, t+1) = E(S, t)\frac{(\bar{f} + c\bar{f})}{\bar{f}} = (1+c)E(S, t) = (1+c)^t \cdot E(S, 0) \tag{19}$$

Formula (19) shows that the operation of selecting schema of which fitness value is higher than the average fitness value of population shows up as exponential growth. As with BGA, MO-GA does not generate new schemas but increases the number of existing schemas with high fitness.

In order to illustrate the effect of crossover on schema in MO-GA, first, we take BGA with single point crossover, for example. Provided that there is a specified string D whose length is seven, two representative schemas S_1 and S_2 contained in string A are as follows:

$$D = 0\ 1\ 1\ 1\ 0\ 0\ 0$$
$$S_1 = *\ 1\ *\ *\ *\ *\ 0$$
$$S_2 = *\ *\ *\ 1\ 0\ *\ *$$

Provided that string D is selected to participate in crossover operation, crossover point is generated randomly. For example, if the crossover point is between the third and fourth positions of D, representing them as "|", we show the effect of crossover on schema S_1 and S_2 as

$$D = 0\ 1\ 1|1\ 0\ 0\ 0$$
$$S_1 = *\ 1\ *|*\ *\ *\ 0$$
$$S_2 = *\ *\ *|1\ 0\ *\ *$$

Obviously, schema S_1 will be destroyed if the crossover object of D is different from D in the determinate position of schema S_1. However, for the same crossover position, schema S_2 will be kept in a string of offspring. So the schema S_1 is more difficult to survive than S_2. Because the crossover position is generated randomly, the probabilities that each crossover position is generated randomly are equal. Since $l(S_1) = 5$ and $l(S_2) = 1$, the probabilities that S_1 and S_2 are destroyed are respectively 5/6 and 1/6. Generally, the probability that schema S is destroyed in BGA is given by

$$p_1 = \frac{l(S)}{s_l - 1} \tag{20}$$

where $l(S) \le s_l - 1$, s_l is string length.

In MO-GA, the number of offsprings is α times the number of their parents according to (14). Therefore, a pair of parents generates 2α offsprings. In order to generate 2α offsprings, both parents must be crossed at stochastic crossover position α times. Suppose that the chromosomes of one of the parents contain schema S. The probability that schema S is destroyed in MO-GA is given by

$$p_2 = \left[\frac{l(S)}{s_l - 1}\right]^{\alpha} \tag{21}$$

where $\doteq(S) \le l - 1$, and s_l is the string length.

The probability that schema S is destroyed is constantly reduced with increase of α according to (20) and (21). Therefore,

$$\left[\frac{l(H)}{s_l - 1}\right]^{\alpha} < \frac{l(S)}{s_l - 1} \tag{22}$$

where $\alpha \in \{2,3,4,\ldots\}$.

Thus,

$$p_2 < p_1 \tag{23}$$

Crossover is a stochastic process, provided that crossover probability is p_c. Then, the survival probability of schema S in MO-GA is given by

$$p_s \ge 1 - p_c \cdot \left[\frac{l(S)}{s_l - 1}\right]^{\alpha} \tag{24}$$

Meanwhile, considering the effect of selection and crossover operations on schema in MO-GA, the selection and crossover operations are not related, and the number of schema S in offsprings is given by

$$E_1(S, t+1) \ge E(s, t) \cdot \frac{f(S)}{\bar{f}}\left[1 - p_c\left(\frac{l(S)}{s_l - 1}\right)^{\alpha}\right] \tag{25}$$

Table 1
Simulation results of class 1 with MO-GA and BGA.

Class 1		burma14	wxp20	eil51	kroB100
Known optimal solution		30.8785	24.5222	442.0471	22,141
Average optimal solution	BGA	30.8785	24.6144	444.6790	22,143.5682
	MO-GA	30.8785	24.5222	442.1863	22,141.2733
Average running time	BGA	0.4645	1.5408	20.1255	398.6571
	MO-GA	0.4261	1.3506	17.2092	306.4538
Average number of iterations	BGA	13.7	38.6	302.1	418.60
	MO-GA	8.6	28.3	267.833	388.59
Relative absolute error	BGA	0%	0.3758%	0.5954%	0.0116%
	MO-GA	0%	0%	0.0315%	0.0015%

From formula (22) and (25), we obviously have

$$E(S,t)\cdot\frac{f(S)}{\bar{f}}\left[1-p_c\cdot\left(\frac{l(S)}{s_l-1}\right)^{\alpha}\right] > E(S,t)\cdot\frac{f(S)}{\bar{f}}\left[1-p_c\cdot\frac{l(S)}{s_l-1}\right] \tag{26}$$

where $\alpha \in \{2,3,4,\ldots\}$, $l(S)\le l-1$.

Letting

$$E_2(S,t+1)=E(S,t)\cdot\frac{f(S)}{\bar{f}}\left[1-p_c\cdot\frac{l(S)}{s_l-1}\right] \tag{27}$$

Results in

$$E_1(S,t+1)\ge E_2(S,t+1) \tag{28}$$

According to (22) and (23), the probability that the schema S is destroyed in MO-GA is significantly less than the corresponding probability in the BGA. Therefore, the MO-GA is able to preserve efficiently chromosomes which are excellent currently.

Suppose schema S is a superior (potential) schema whose fitness is higher than the average fitness value of population, formula (28) is shown that the number of schema S in MO-GA is significantly more than BGA at time $t+1$, if the number of schema S are $E(S,t)$ in MO-GA and BGA at time t. therefore, MO-GA has higher efficiency of population evolution and faster speed of convergence.

5.2. Performance analysis based on population competition

For MO-GA, population size is unchangeable at each iteration. However, the number of offsprings is significantly increased as compared to the BGA in order to maintain population size unchanged. In this process, we preserve the outstanding individuals, and eliminate poor individuals. This operation can be called population competition operation. It makes the population more competitive, and speeds up the survival of the fittest. Therefore, MO-GA has faster speed as compared to the BGA.

6. Simulation experiments

In order to verify the effectiveness of MO-GA, simulations were carried out with *Matlab R2013a* in comparison to BGA. The TSP instances that have been chosen for experimentation are classified into two classes based on their problem size. Class1 of instances are considered as small sized and class 2 are considered as large sized TSP instances. Six data sources of TSP were used as follows: (1) wxp20 with twenty cities in reference [22], (2) burma14 in TSPLIB with fourteen cities, (3) eil51 in TSPLIB with fifty one cities, (4) kroB100 in TSPLIB with one hundred cities, (5) pr1002 in TSPLIB with one thousand and two cities, (6) pcb3038 in TSPLIB with three thousand and thirty eight cities.

The MO-GA and BGA programs were run one hundred times, and the averages of the results were calculated. In the experiments, the parameters were chosen as mutation rate = 0.3, and population size = 100. The iterative termination condition of class 1 is that the optimal solution obtained by the MO-GA and BGA is less than or equal to the known optimal solution, or max number of generations equal to 1000. The iterative termination condition of class 2 is that the max running time is 7200s. For the six TSP, the average computing time and average number of iterations are shown in Tables 1 and 2.

Table 2
Simulation results of class 2 with MO-GA and BGA.

Class 2		pr1002	pcb3038
Known optimal solutions		259,045	137,694
Average optimal solutions	BGA	271,896.462	147,458.263
	MO-GA	265,136.231	142,310.123
Average computation time	BGA	1983.751	423.529
	MO-GA	1376.254	326.667
Relative absolute error	BGA	4.9611%	7.0913%
	MO-GA	2.35142%	4.0786%

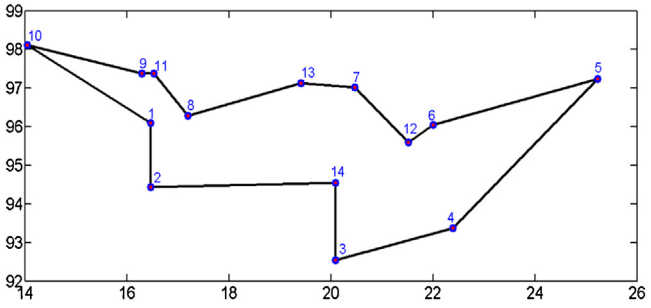


Fig. 3. The best route of burma14.

The average computing time and the average number of iterations in Table 1 were calculated as follows:

When the iteration termination condition is satisfied, the number of iterations and time of processing at the ith run are respectively $niter(i)$ and $t(i)$, $i=1,2,\ldots,k$, where k is the number of runs used in each experiment. Then, the average computing time and average number of iterations are computed by

$$t_{av}=\frac{1}{k}\sum_{i=1}^{k}t(i) \tag{29}$$

$$niter_{av}=\frac{1}{k}\sum_{i=1}^{k}niter(i) \tag{30}$$

The best routes of the six examples are plotted in Figs. 3–8, respectively.

Table 1 shows that MO-GA reduces the average running time by 11.2979% for burma14, 12.3442% for wxp20, 14.906% for eil5 and 23.1285% for kroB100 in comparison to BGA. Similarly, the number of iterations are reduced by 37.2263% for burma14, 26.6839% for wxp20, 11.3429% for eil5, and 7.1691% for kroB1002.

Table 2 shows that MO-GA reduces the average number of iterations by 30.6237% for pr1002, and 32.4411% for pcb3038. In

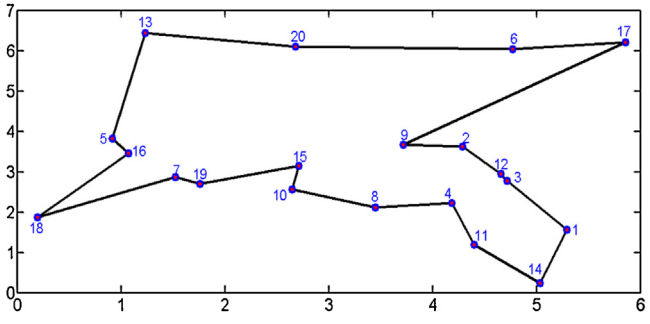


Fig. 4. The best route of wxp20.

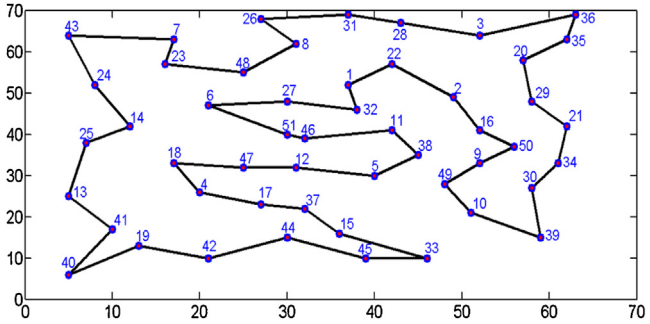


Fig. 5. The best route of eil51.

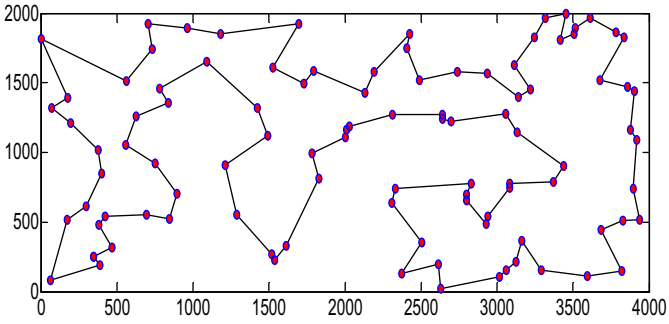


Fig. 6. The best route of kroB100.

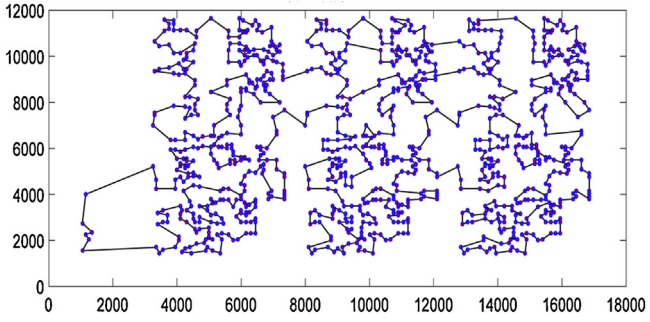


Fig. 7. The best route of pr1002.

addition, the solved optimal solution of MO-GA is superior to that of BGA.

In summary, the simulation results with the six TSP examples show that MO-GA has faster calculation speed and reduced number of iterations as compared to BGA.

Tables 1 and 2 also show that MO-GA reaches a deeper minimum or the same minimum of the cost function as compared to BGA.

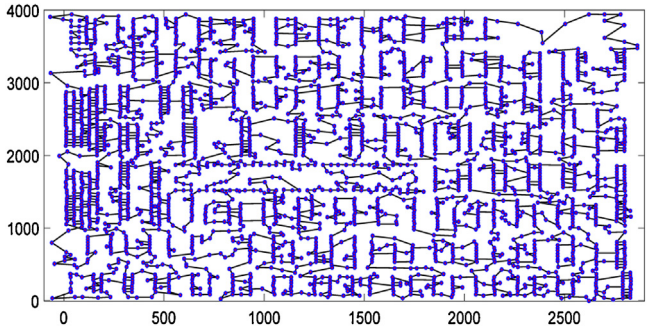


Fig. 8. The best route of pcb3038.

7. Conclusions and expectation

MO-GA inspired by biological evolutionary and mathematical ecological theories was introduced. The evolutionary strategy of MO-GA to generate multi number of offsprings was discussed in terms of crossovers and mutations used. In addition, a novel method of computing fitness value was discussed. The method achieves fast convergence and rich population diversity as well as convergence to a deep minimum of the cost function.

The basic difference between MO-GA and BGA is the number of offsprings generated with the crossover operator and the mutation operator. With the crossover operator of MO-GA, this is $2n$ (or more), and in BGA it is n. With the mutation operator of MO-GA, this is $2np_m$, and in BGA it is np_m. The other properties are the same. More number of offsprings increases the possibility of producing excellent members of the population, thus enabling faster calculation speed, less number of iterations and typically reaching a deeper minimum of the cost function as compared to BGA.

MO-GA has many issues worth further studying, for example, the method of generating multi-offsprings with different crossover methods, the relationship between number of offsprings and computing speed, and so on.

Acknowledgment

The paper is supported by sub project of National Science and Technology support program under Grant 2014BAD06B01-23, and the young backbone teachers project of China Scholarship Council (File No. 201306615012).

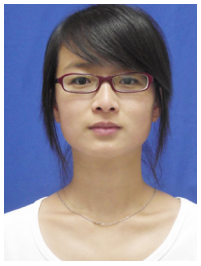
References

[1] A. Philip, A.A. Taofiki, O. Kehinde, A genetic algorithm for solving travelling salesman problem, Int. J. Adv. Comput. Sci. Appl. 2 (1) (2011) 26–29.
[2] M.R. Graey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman W.H., San Francisco, 1979.
[3] C.H. Papadimitriou, K. Stegilitz, Combinatorial Optimization: Algorithms and Complexity, Prentice Hall of India Private Limited, India, 1997.
[4] S. Singh, E.A. Lodhi, Study of variation in TSP using genetic algorithm and its operator comparison, Int. J. Soft Comput. Eng. 3 (2) (2013) 2231–2307.
[5] S. Lin, Computer solutions of the traveling salesman problem, Bell Syst. Tech. J. 10 (10) (1965) 2245–2269.
[6] D. Whitely, T. Starkweather, F.D. Ann, Scheduling problems and traveling salesman: the genetic edge recombination operator, in: Proceedings of the 3rd International Conference on Genetic Algorithms, 1989, pp. 133–140.
[7] S. Kirkpatrick, C.D. Gelatt, M.P. Vechi, Optimization by simulated annealing, Science, New series 220 (4598) (1983) 671–680.
[8] F. Alizadeh, R.M. Karp, R.M. Newberg, et al., Physical mapping of chromosomes: a combinatorial problem in molecular biology, in: SODA' 93 Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms, 1993, pp. 371–381.
[9] C. Korostensky, G.H. Gonnet, Using traveling salesman problem algorithms for evolutionary tree construction, Bioinformatics 16 (7) (2000) 619–627.
[10] B.D. Corwin, A.O. Esogbu, Two machine flow shop scheduling problems with sequence dependent setup times: a dynamic programming approach, Naval Res. Logist. Q. 21 (3) (1974) 515–524.

[11] H. Gao, B. Feng, L. Zhu, Reviews of the meta-heuristic algorithms for TSP, Control Decis. 21 (3) (2006) 241–247, 252.
[12] D.E. Goldberg, J.R. Lingle, Alleles, loci and the traveling salesman problem, in: Proceedings of the First International Conference on Genetic Algorithms and Their Applications, Pittsburgh P A Carnegie Mellon University, Pittsburgh, 1985, pp. 154–159.
[13] J.J. Grefenstette, R. Gopal, B. Rosmaita, et al., Genetic algorithm for TSP, in: Proceedings of the First International Conference on Genetic Algorithms and Their Applications, Pittsburgh P A Carnegie Mellon University, Pittsburgh, 1985, pp. 160–168.
[14] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys, The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization, Wiley, Chichester, 1985.
[15] P. Larranaga, C. Kuijpers, R. Murga, et al., Genetic algorithms for the traveling salesman problem: a review of representations and operators, Artif. Intell. Rev. 13 (2) (1999) 129–170.
[16] X. Mou, D. Xie, W. Yan, Research based on genetic algorithm traveling sealer problem of trajectory optimization, J. Syst. Simul. 25 (Suppl.) (2013) 86–89.
[17] L. Wang, X. Niu, Application genetic algorithms for dynamic TSP, Inf. Commun. 123 (2013) 35.
[18] Q. Wang, D. Yuan, D. Liang, A greedy GA of traveling salesman problem, Manuf. Autom. 35 (1) (2013) 71–74.
[19] Y. Wei, M. Zhao, X. Huang, A novel greedy genetic algorithm for traveling salesman problem, Comput. Eng. 30 (19) (2004) 19–20, 34.
[20] D. Wang, J. Wang, H. Wang, et al., Intelligent Optimization Method, Higher Education Press, Beijing, 2007.
[21] F. Wang, J. Wang, C. Wu, et al., The improved research on actual number genetic algorithm, J. Biomath. 21 (1) (2006) 153–158.
[22] X. Wang, L. Cao, Genetic Algorithm-Theory, Application and Software Implementation, Xi'an Jiaotong University Press, Xi an, 2005.

Jiquan Wang received the BE degree in Agricultural electrification and automation from the Northeast Agricultural University, China, in 1996, the ME degree in Agricultural electrification and automation from the Northeast Agricultural University, China, in 2004, and the Ph.D. degree in Agricultural Equipment Engineering Technology from Shenyang Agricultural University, China, in 2011. He is currently an associate professor with Engineering College, Northeast Agricultural University. His current research interests include genetic algorithm theory and its application, neural Network theory and its application, parallel computing and image recognition. He has published over 40 papers in domestic and international academic journals and conference proceedings.

Okan K. Ersoy received the BSEE from Robert College in 1967, the MSEE degree from University of California at Los Angeles in 1968, Ph.D. degree from University of California at Los Angeles. He is currently a professor of Electrical and Computer Engineering, Purdue University. His current research interests include neural networks, image processing and imaging, networking and information processing, genetic algorithms, decision trees and support vector machine machines. He has published over 200 papers in international academic journals and conference proceedings.



Mengying He received the BE degree in Industry Engineering, Northeast Agricultural University, China in 2013. She is currently a postgraduate student of Engineering College, Northeast Agricultural University. Her current research interests include genetic algorithm theory and its applications.

Fulin Wang received the BE degree in agricultural mechanization and automation from Northeast Agricultural University, China, in 1981, the ME degree in agricultural mechanization and automation from Northeast Agricultural University, China, in 1988, and the Ph.D. degree in agricultural equipment engineering technology from Shenyang Agricultural University, China, in 1993. He is currently a professor with the Engineering College, Northeast Agricultural University. His current research interests include genetic algorithm theory and its applications, neural network theory and its applications, parallel computing and image recognition. He has published over 100 papers in domestic and international academic journals and conference proceedings.