Hindawi
Journal of Control Science and Engineering
Volume 2022, Article ID 4398952, 10 pages
https://doi.org/10.1155/2022/4398952

Research Article

An Improved Directed Crossover Genetic Algorithm Based on Multilayer Mutation

Feng Xie,[1] Quansheng Sun,[2] Yinfeng Zhao,[2] and Haibo Du[2,3]

[1]*China Energy Engineering Group, Anhui Electric Power Design Institute Co.LTD, Hefei, China*
[2]*School of Electrical Engineering and Automation, Hefei University of Technology, Hefei, Anhui 230009, China*
[3]*Anhui Province Engineering Research Center for Industrial Automation, Hefei University of Technology, Hefei, Anhui, China*

Correspondence should be addressed to Haibo Du; haibo.du@hfut.edu.cn

Received 17 February 2022; Revised 13 June 2022; Accepted 15 June 2022; Published 19 September 2022

Academic Editor: Radek Matušů

In order to solve the shortcomings of traditional genetic algorithms in image matching in terms of computational speed and matching accuracy, this paper proposes a directed crossover genetic matching algorithm (DCGA) based on multilayer variation. The algorithm differs from the traditional genetic algorithm (GA) in which the crossover strategy is improved and a multilayer adaptive variation operator is introduced. The crossover operation selects a certain proportion of spherical individuals from each generation as the evolutionary target, and the rest of the individuals evolve towards it in each dimension; the variation operation stratifies the population and adopts different adaptive variation methods for different layers. Avoiding the shortcomings of traditional genetic algorithms that tend to fall into local extremes, thus alleviating premature convergence, improves the search performance of the algorithm. The algorithm proposed in this paper is compared with the commonly used genetic algorithm by testing the effect of the function and tested practically in template matching. The experimental results show that the improved genetic algorithm has better convergence speed and search accuracy.

1. Introduction

Image matching includes template matching, target matching, and dynamic pattern matching, among which template matching is the most common matching method, which is widely used in remote sensing images, medical imaging, computer vision, and many other fields. Traditional matching algorithms adopt an iterative region-dependent matching search strategy, which results in a large amount of time spent on nonoptimal matching points, making it difficult to use in real-time applications. Therefore, some foreign scholars have started to use genetic algorithms to study template matching problems, such as literature [1]. A large amount of practice has shown that traditional genetic algorithms applied to image matching may have the following shortcomings: (1) genetic operations tend to destroy the construction of individuals during recombination, resulting in the algorithm approaching a local optimum or premature maturity; (2) the computational effort of adaptation is large, resulting in the algorithm running at a speed that cannot meet the system's real-time needs. Therefore, the genetic algorithm needs to be improved when applied to image matching.

There are two types of fast image matching algorithms in practice: the sequential similarity detection algorithm [2] (SSDA) and the multiresolution tower structure algorithm [3] (MPSA). The SSDA algorithm guarantees global optimality in image matching, but the SSDA algorithm has a variable matching time per frame with different target positions, which affects its implementation in practical engineering. The MPSA algorithm requires essentially the same matching time per frame, but its algorithm has the possibility of mismatching in principle, especially in low contrast conditions where it is more likely to be lost, and the algorithm trades the matching accuracy loss for speed improvement.

Heuristic algorithms are often used to solve optimization problems in real life, and there are three main categories:

genetic algorithm, simulated annealing [4], and particle swarm optimization [5]. Compared with the other two algorithms, the genetic algorithm has the characteristics of simple and general, strong robustness, suitable for parallel processing, and high efficiency. Genetic algorithms take all individuals in the population as objects and use randomization techniques to guide an efficient search of an encoded parameter space. As Holland of the University of Michigan proposed [6], more and more researches have been carried out around this algorithm. After years of development, genetic algorithms are widely used in power grid line loss calculation [7], solving the traveling salesman problem [8], circuit design optimization [9], cloud computing resource allocation [10], job shop scheduling [11, 12], and other aspects, the algorithm is often used for the optimal solution of multipeak functions.

The main purpose of this paper is to propose a new directional crossover genetic algorithm by improving the crossover strategy and introducing the idea of multilayer adaptive mutation operator, so as to solve the problems of slow convergence and premature convergence of the genetic algorithm. The main contributions of this paper are as follows: (1) Directed crossover operator: directed crossover is an improvement to the crossover operation of traditional genetic algorithms, using the current optimal solution as the evolutionary direction of the target population, which is more responsive to the evolutionary direction of the whole population. Experimental results show that the directed crossover operator has faster convergence and higher search accuracy than the traditional genetic algorithm with a finite parent crossover operator. (2) The introduction of multilayer adaptive mutation operator: at the same time, the hierarchical adaptive mutation operator is proposed, and three kinds of mutation scale are introduced. The algorithm can adjust the variation range according to the convergence, ensuring that the algorithm can still explore the rest of the solution space while converging.

2. The Principle of Traditional Genetic Algorithm

In the genetic algorithm, n-dimensional decision vector $X = [x_1, x_2, \ldots, x_n]^T$ is represented by a symbol string X composed of n symbols $x_i\,(i = 1, 2, \ldots, n)$: $X = x_1, x_2, \ldots, x_n$ regards each x_i as a genetic gene, and all its possible values are called alleles. In this way, X can be regarded as a chromosome composed of n genetic genes. The simplest allele is composed of two integers 0 and 1, and the corresponding chromosome can be represented as a string of binary symbols. The permutation form X formed by this code is the genotype of the individual, and the X value corresponding to it is the phenotype of the individual. Chromosome X is also called individual X. For each individual X, its fitness should be determined according to certain rules. The traditional genetic algorithm encodes the possible solutions of the target problem as chromosomes x_i, the target problem is abstracted into a fitness calculation function $f(x_i)$, and the higher the fitness, the closer the chromosome is to the optimal solution of the problem. Multiple chromosomes constitute a population, and through the steps of crossover, mutation, and selection, the population converges to the optimal solution in multiple iterations.

At the beginning of the algorithm, the population $X = \{x_1, x_2, \ldots, x_n\}$ is randomly generated, and the fitness of the population is calculated. Then, enter the selection link, randomly remove a part of the chromosomes according to the fitness, and supplement with the same amount of individuals with high fitness to form a new population. Then, the crossover operator is performed, the crossover probability is set to p_c, the parent x_i and x_j cross to generate the child $\overline{x}_i$ and $\overline{x}_j$, and the formula is as follows:

$$\begin{aligned} \overline{x}_i &= (1-\alpha)x_i + \alpha x_j, \\ \overline{x}_j &= (1-\alpha)x_j + \alpha x_i. \end{aligned} \tag{1}$$

Here, α is a random number between (0, 1), which reflects the inheritance degree of the offspring to the parent traits. Then, mutate individual x_i to generate x_{i+1}. The mutation formula is as follows:

$$x_{i+1} = \begin{cases} x_i + x_i \times \text{rand}\, n & t \le p_m, \\ x_i, & t > p_m, \end{cases} \tag{2}$$

where rand $n()$ represents the random number in the interval $(-1, 1)$, t is the random number between $(0, 1)$, and p_m represents the probability of mutation.

Cyclic execution of crossover, mutation, and selection operators can make the species chromosomes approach the optimal solution gradually, until the evolutionary algebra of the population reaches the specified value or the chromosome with the highest fitness meets the requirements. At this time, the best individual is the target searched by the algorithm. The traditional genetic algorithm is shown in Algorithm 1.

3. Directed Crossover Genetic Algorithm for Multilayer Mutation

Most of the traditional genetic algorithms use the method of limited parental crossover, which is difficult to reflect the overall situation of the population. It has the shortcomings of slow convergence speed and easy to fall into the extreme value, and the mutation operator has a single structure, resulting in poor search ability in the early operation of the algorithm and low search accuracy in the later operation. Low problem: in this paper, the idea of directional crossover is introduced, and the directional crossover operator of the algorithm is improved. At the same time, the hierarchical adaptive mutation operator is optimized, and a directional crossover genetic algorithm based on multilayer mutation is constructed.

3.1. Directional Crossing. The crossover process of the traditional genetic algorithm often adopts the crossover of two parents. In order to make the population converge faster, researchers have proposed some methods of multiparent crossover, such as the three-parent crossover operator

Input: $f(x_k)$:fitness function; X:population; x_k:parents; $\overline{x}_k$:children; n:numbers of evolutions; N:the predetermined number of iterations;
Output: optimal individual X
1: initialize X, $n = 0$ and N;
2: **repeat**
3: evaluate fitness function $f(x_i)$ of population x_i
4: choose the population x_i with higher fitness
5: select operation to x_i according to formula (1)
6: Mutation operation to x_i according to formula (2)
7: **until** $n > N$

Algorithm 1: Traditional genetic algorithm.

mentioned in literature [13, 14], from three-parent chromosomes The individual with the greatest fitness is selected as the direction of the crossover. The idea of these crossover operators is to crossover within the local range limited by the parent, and the evolution range is limited. In order to better reflect the evolutionary direction of the population as a whole during the crossover process, this paper introduces the idea of directional crossover and sorts the chromosomes in the population according to the fitness. The greater the fitness, the better the chromosomes. In each iteration, the optimal part of individuals is selected as the common parent of the offspring.

Assuming that, after the selection operation, the individuals of the a-th generation population are sorted as $X_a = \{x_a^1, x_a^2, \ldots, x_a^n\}$ according to their fitness size. For the D-dimensional problem, a single chromosome $x_a^i = [x_{a,1}^i, \ldots, x_{a,D}^i]$, and the population size is n. Let the number of selected excellent chromosomes be s, and the remaining $n - s$ chromosomes of the population are crossed with them, respectively. The formula of the crossover operator is as follows:

$$\overline{x}_{a,j}^i = x_{a,j}^i + \frac{f_d}{f_i + f_d} \times \left(x_{a,j}^d - x_{a,j}^i\right). \tag{3}$$

In formula (3), $\overline{x}_{a,j}$ represents the j-th dimension chromosome generated by crossover, $x_{a,j}^i$ represents the less adaptive chromosome in the a-th generation, $x_{a,j}^d$ represents the excellent chromosome in this generation, and f_i, f_d represent the fitness of individuals $x_{a,j}^i$ and $x_{a,j}^d$. In the crossover process, the traits inherited by the offspring are affected by the fitness of the two parents, that is, the outstanding ones in the parents can pass on more imprints to the next generation, which improves the convergence speed. The directional crossover process is shown in Figure 1.

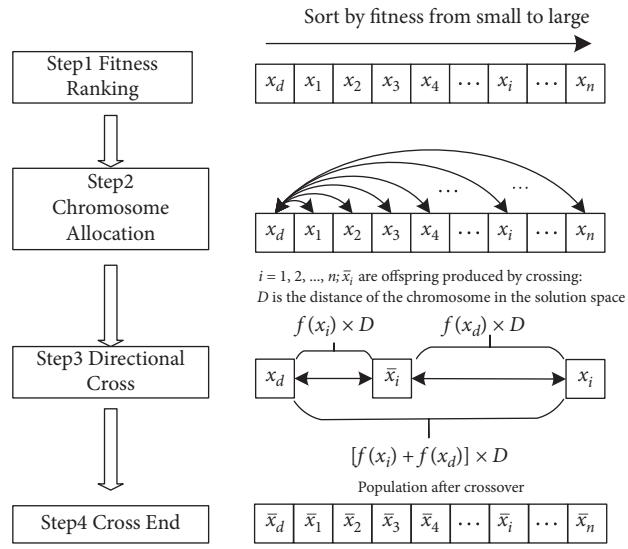

Figure 1: Schematic diagram of the directional crossover process.

3.2. Hierarchical Adaptive Mutation. In the binary coded traditional genetic algorithm, the mutation operator flips every gene in chromosome with the same mutation probability. In the literature [15], the authors have proved that traditional mutation methods cannot effectively maintain the diversity of alleles at the same locus. In order to solve the shortcomings of traditional mutation operators with single structure and small mutation range, this paper proposes a corresponding hierarchical adaptive mutation operator that adjusts the mutation range based on convergence conditions.

Literature [16] proposed the concept of hierarchical mutation and improved the variation operation of traditional genetic algorithms by introducing the concept of hierarchical variation. Literature [17–19] shows that the convergence speed and accuracy of the algorithm can be well optimized by improving the variational operation. In this regard, this paper adopts hierarchical adaptive mutation and divides the population into four layers according to the size of fitness, and the number of chromosomes in each layer is recorded as A, B, C, D. For individuals with different fitness, different mutation methods are applied to achieve the diversification of population search functions. Among them, layer A stores the current optimal chromosomes and does not participate in mutation, which can protect excellent traits.

In layer B, individuals with similar fitness order between two generations are selected to represent the evolution direction of the population. Preliminary simulation shows that group B represents individuals with similar fitness to the optimal parent generation in the operation process. The

convergence goal is clear, and the intergenerational difference can effectively increase the performance of fine search. The mutation operator is as follows:

$$x_{a+1,j}^{i} = \overline{x}_{a,j}^{i} + \text{rand}\, n() \times \left(\overline{x}_{a,j}^{i} - \overline{x}_{a-1,j}^{i}\right), \tag{4}$$

where C is the j-th generation generated after mutation; $\overline{x}_{a,j}^{i}, \overline{x}_{a-1,j}^{i}$ are the same dimension of the current generation chromosome and the previous generation chromosome with the same population fitness order, respectively; rand n represents the standard normal distribution with the mean value of 0 and variance of 1.

In layer C, adaptive mutation strategy was adopted according to the current search range of the population. Because the fitness of the C-layer chromosome occupies a medium position in the population, it has limited role in solving the current exact solution, so it undertakes a large range of search work. The search area adopts an adaptive strategy, adjusting itself according to the current population dispersion. The formulas are as follows:

$$\begin{aligned} x_{a+1,j}^{i} &= \overline{x}_{a,j}^{i} + \text{norm}rnd(0, l), \\ l &= k \times \frac{\left(X_{\max} - X_{\min}\right) - \left(\overline{x}_{a,j}^{\max} - \overline{x}_{a,j}^{\min}\right)}{X_{\max} - X_{\min}}, \end{aligned} \tag{5}$$

where norm$rnd(0, l)$ represents a random number with a mean of 0 and a variance of 1. $[X_{\min}, X_{\max}]$, $[\overline{x}_{a,j}^{\min}, \overline{x}_{a,j}^{\max}]$, respectively, represent the constraint range solved by the algorithm and the search range of the j-th dimension of the current population. k is the scaling factor. It can be seen from the formula that the variance l is positively correlated with k, and the variation effect of this grouping can be changed by adjusting k. When k is too large, the variation tends to be random, and the convergence rate slows down. If k is too small, the search range is narrow, and the algorithm converges prematurely. Through experiments, the convergence speed and search performance can be balanced when k is between $(2, 3)$. It can be seen that when the population convergence degree is high, the standard deviation l becomes larger, and the search range after C-layer variation increases. When the population is dispersed, l is smaller, resulting in rapid convergence of chromosomes. This mutation operator has a feedback control effect on the global convergence.

Layer D has the smallest fitness, so it has the largest range of variation and is responsible for completely random search tasks.

$$x_{a+1}^{i} = x_{a}^{i} + P \times \left(X_{\max} - X_{\min}\right). \tag{6}$$

$P \in (-0.4, 0.4)$ is random number. Large scale variation occurred in this group of individuals, and the individuals after variation were randomly distributed in half solution space, further expanding the search scope. For this reason, intergenerational detection is set. The algorithm detects the search results every t generation. When the fitness increase of the current optimal solution is found to be small, random initialization is performed on the chromosomes of layers B, C, and D.

3.3. Algorithm Flow. The algorithm in this paper adopts the termination method of the specified evolutionary algebra, and the overall flow of the improved algorithm is shown in Figure 2. Algorithm 2 is the framework of the improved algorithm in this paper.

In addition to the crossover operator and mutation operator mentioned in the text, the selection operator is also used in the algorithm to simulate the phenomenon of survival of the fittest in nature. In the selection, this paper adopts the roulette algorithm [20]: assuming that the fitness of individual x_i is $f(x_i)$, and the number of individuals contained in the population is n, the probability of it being retained is expressed as

$$p(x_i) = \frac{f(x_i)}{\sum_{k-1}^{n} f(x_k)}, \tag{7}$$

and this selection method can ensure that all individuals have a certain probability of being retained, and the greater the fitness, the greater the probability of being retained.

4. Experiments and Results Analysis

In order to verify the effectiveness of the designed directional crossover genetic algorithm based on multilayer mutation, eight tested functions are selected and compared with the three genetic algorithms of IGA [21], IRCGA [22], and TAGA [23], and the accuracy and convergence speed of the search results are compared.

4.1. Comparing Examples and Test Functions. In tests, the population size of each algorithm is set to 100, the number of iterations is 100, each algorithm runs independently 20 times, and the search range x_i of the population limit is $(-10, 10)$. The four algorithms to be tested are as follows:

DCGA: the directional cross genetic algorithm is proposed in this paper; the algorithm parameter settings are introduced in Section 2: $s = 5, A = B = 20, C = D = 30, t = 5$.

IGA is an improved genetic algorithm based on real number coding, which is a classic algorithm that achieves adaptive adjustment during operation by changing the crossover and mutation probability. Mutation parameters $r = 0.2, b = 3$; adaptive parameters $k_1 = k_2 = 1, k_3 = k_4 = 0.5$.

IRCGA: the improved real-coded genetic algorithm adopts the strategy of adaptively changing p_c and p_m and performs multiparental crossover, which can theoretically increase the speed of convergence and increase the efficiency of evolution: $p_1 = 0.65, p_2 = 0.2, c_1 = c_2 = 1.5$.

TAGA: an optimized genetic algorithm proposes the idea of learning from the best individual: $p_c = 0.65, p_m = 0.2, a_i \in (-0.5, 1.5), c_1 = c_2 = 1.5$.

We choose the following eight test functions:

Bohachevsky function: it is a unimodal bowl function and slowly converges to the minimum value 0 at [0, 0].

$$f = 0.7 + x^2 + 2y^2 - 0.3\cos(3\pi x) - 0.4\cos(4\pi y). \tag{8}$$

Schaffer function: it is a multimodal function and gets the minimum value −1 at [0, 0].

Input: $f(x)$:fitness function; X:crossed population; gen:number of evolutions;
Output: mutated population
Initialize: the population X is ranked $\{x_1, \ldots, x_n\}$ according to fitness;
if $0 < x\_d < A$ **then** then
    no mutation; **return**
else if $f(A) < f(x\_d) < f(B)$ **then**
    do B-layer mutation operation; **return**
else if $f(B) < f(x\_d) < f(C)$ **then**
do C-layer mutation operation; **return**
elsedo D-layer mutation operation; **return**
end if
if gen can be divided by 5 **then**
    if fitness(gen) − fitness(gen − t) < 0.001 **then**
        initialize layer B,C and D; **return**
    end if
end if

Algorithm 2: Hierarchical adaptive mutation algorithm.

$$f = -0.5 + \frac{\left[\sin\left(x^2 + y^2\right)\right]^2 - 0.5}{\left[1 + 0.001 \times \left(x^2 + y^2\right)\right]^2}. \tag{9}$$

Griewank function: it is a multimodal function and gets the minimum value 0 at [0, 0].

$$f = \frac{x^2 + y^2}{4000} - [\cos(x) + 1] \times \left[\cos\left(\frac{y}{\sqrt{2}}\right) + 1\right]. \tag{10}$$

DropWave function: it is a multimodal function and gets the minimum value −1 at [0, 0].

$$f = -\frac{1 + \cos\left(12\sqrt{x^2 + y^2}\right)}{0.5\left(x^2 + y^2\right) + 2}. \tag{11}$$

Rastrigin function: it is a multimodal function which has multiple local extremum and gets the minimum value 0 at [0, 0].

$$f = 20 + \left[x^2 - 10\cos(2\pi x)\right] + \left[y^2 - 10\cos(2\pi y)\right]. \tag{12}$$

Sum-squares function: it is a unimodal function and gets the minimum value 0 at [0, 0].

$$f = x^2 + 2y^2. \tag{13}$$

Booth function: the function is disc-shaped and converges more slowly and gets the minimum value 0 at [1, 3].

$$f = (x + 2y - 7)^2 + (2x + y - 5)^2. \tag{14}$$

Levy-13 function: this function is a multimodal function with a lot of folds and gets the minimum value 0 at [1, 1].

$$\begin{aligned} f = {} & \sin^2(3\pi x) + (x - 1)^2\left[1 + \sin^2(3\pi y)\right] \\ & + (y - 1)^2\left[1 + \sin^2(2\pi y)\right]. \end{aligned} \tag{15}$$

4.2. Results of Algorithm Experiment and Analysis. This paper takes the error between the searched value and the theoretical maximum value as the main comparison data. For the same test function, each algorithm is run twenty times. The test results are shown in Table 1. The convergence of each algorithm under each test function is shown in Figure 3. The minimum mean and minimum standard deviation in each test function are bolded in the figure.

Since most of the test functions achieve the minimum value at [0, 0], in order to prevent the special value of 0 from interfering with the running effect of the algorithm, before running, all test functions are shifted by 2.5 unit lengths in each dimension to reflect the randomness of the solution.

The local extremum of the multimodal function Schaffer is ring-shaped, and there are continuity equivalent solutions at the folds, which can easily interfere with the algorithm and make the algorithm converge on the extremum ring. It can be seen from the experimental results that the IGA, IRCGA, and TAGA algorithms are all trapped at the extreme value solution of 0.99, while the DCGA algorithm in this article can further jump out of the extreme value and find the most value point.

For Griewank, DropWave, Rastrigin, and Levy-13, the extreme point of this kind of multimodal function presents a peak shape around the extreme point, and the genetic operator can jump out of the extreme point. This type of function tests the convergence performance of the algorithm's crossover operator. It can be seen from the iterative error evolution curve that the new algorithm has the fastest convergence speed in the multimodal function, and the accuracy is much higher than other algorithms. Once a possible solution is found, the algorithm can lock until the mutation operator makes the population find a better target.

The unimodal functions Bohachevsky, sum-squares, and booth have a small slope at the peak and a gentle peak, which are suitable for the search accuracy of the detection algorithm. The new algorithm in this paper shows excellent performance for these problems, fast convergence, and small error. The other algorithms slow down their convergence under a flat objective function, resulting in greater error than the DCGA algorithm.

It can be seen from Table 1 that, for the eight test functions, the errors of the DCGA algorithm are much

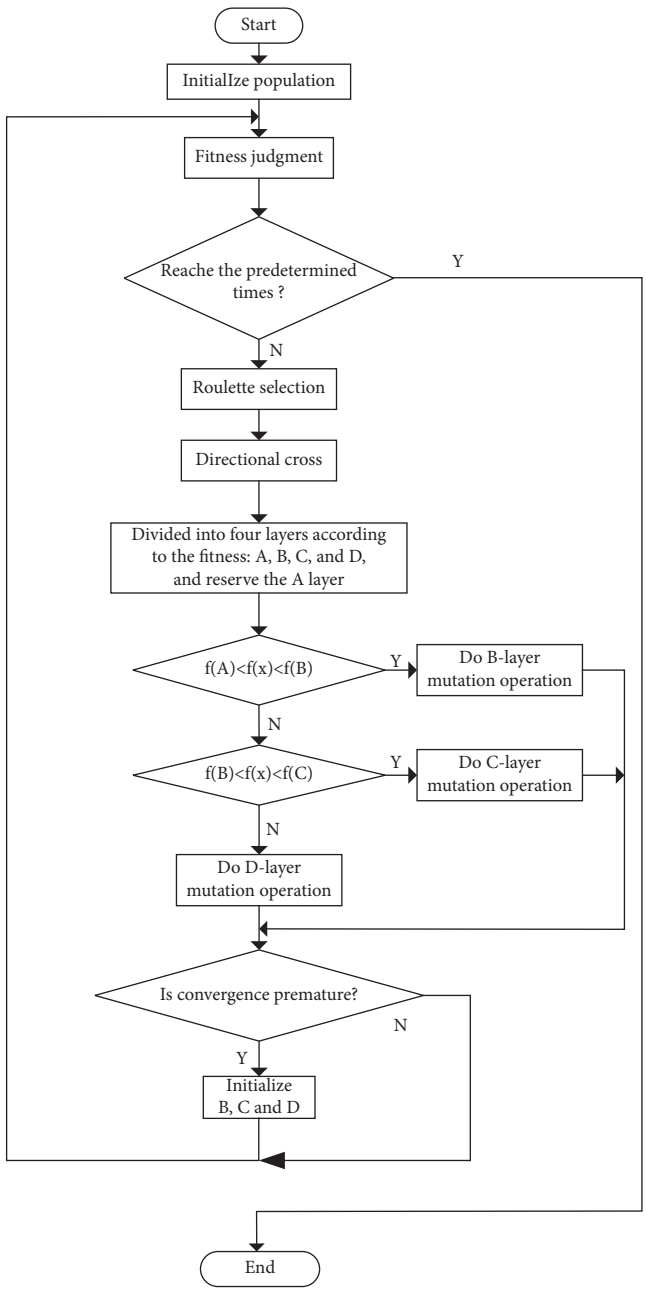

Figure 2: Flowchart of the adaptive directional crossover genetic algorithm.

smaller than that of the control group, and the smaller standard deviation also indicates that the new algorithm is also very stable in multiple runs.

4.3. Algorithms Tested in Template Matching. Template matching is to search for the target image position in the original image. When searching, it is necessary to take a subimage from the original image and compare it with the template. If the similarity between the two reaches a certain level, it can be considered that the target position has been found, and the search is over.

Assuming that the template is S and the size is $M \times N$ pixels, we place the template on the original image and move it. The coordinate (i, j) in the upper left corner represents the current position. When the template moves, the corresponding subimage T of the same shape is captured in the original image, and the subimage T is used. The Euclidean distance with the template S is the basis for judging the similarity:

$$D(i,j) = \sum_{m=1}^{M} \sum_{n=1}^{N} \left[S^{ij}(m,n) - T(m,n)\right]^2. \tag{16}$$

In formula (16), the smaller the D is, the higher the similarity between S and T is. In order to facilitate comparison, formula (16) is normalized, and the following formula is obtained after processing:

$$R(i,j) = \frac{\sum_{m=1}^{M} \sum_{n=1}^{N} S^{ij}(m,n) \cdot T(m,n)}{\sqrt{\sum_{m=1}^{M} \sum_{n=1}^{N} \left[S^{ij}(m,n)\right]^2} \sqrt{\sum_{m=1}^{M} \sum_{n=1}^{N} \left[T(m,n)\right]^2}}. \tag{17}$$

In formula (17), the similarity R represents the matching degree between the subgraph and the template under the corresponding coordinates, and the closer the R is to 1, the higher the matching degree. Considering the similarity calculation formula as a fitness function, the template matching problem can be regarded as a two-dimensional function optimization problem.

The template used in the test and the original image are shown in Figure 4. The search template is in the white box, the size is $88 * 98$ pixels, and the size of the original image is $1024 * 1024$ pixels.

In this paper, the coordinates of the upper left corner point (i, j) are used as independent variables. The optimal solution for matching is the pixel point $(586, 580)$. We simulate matching algorithm based on DCGA on the MATLAB software platform and compare the results with the runs of SSDA, traditional genetic algorithm (GA), and adaptive genetic algorithm (AGA) [24]. The DCGA algorithm proposed in this paper uses an improved directed crossover operation compared to the AGA algorithm and a hierarchical adaptive variation operator compared to the traditional GA algorithm. The population size and the number of iterations were set to 100, each algorithm was run independently for 50 times, and the similarity between the average searched subgraph and the template and the average number of iterations when convergence was recorded was recorded. The experimental results are shown in Table 2.

It can be seen from Table 2 that compared with the SSDA algorithm, the genetic algorithm can reduce the image matching time to a greater extent, among which the DAGA algorithm proposed in this paper has the shortest time in image matching. To verify the excellent performance of the DAGA algorithm in terms of global convergence, DAGA is compared with the GA and AGA algorithms. In 50 matching tests, the DCGA-based image matching algorithm achieved 100 percent matching accuracy every time, while the average matching accuracy of the GA and AGA algorithms were 79.1 percent and 88.3 percent, respectively, indicating that the algorithm proposed in this paper has good global convergence.

Table 1: Algorithm running results error comparison.

Algorithms	DCGA		IGA		IRCGA		TAGA	
Test functions	Mean	Std. dev	Mean	Std. dev	Mean	Std. dev	Mean	Std. dev
Schaffer	$4.86E-04$	$2.12E-03$	$4.54E-03$	$4.69E-03$	$9.45E-03$	$1.17E-03$	$6.72E-03$	$3.09E-03$
Griewank	$1.99E-08$	$6.51E-08$	$9.40E-04$	$2.27E-03$	$1.93E-03$	$3.16E-03$	$3.02E-03$	$2.38E-03$
DropWave	$8.80E-11$	$2.85E-10$	$4.79E-03$	$1.38E-02$	$4.94E-02$	$2.48E-02$	$5.65E-02$	$2.05E-02$
Rastrigin	$4.86E-09$	$1.93E-08$	$5.68E-02$	$8.04E-02$	$4.52E-01$	$4.60E-01$	$7.45E-01$	$1.63E-01$
Bohachevsky	$7.85E-08$	$2.18E-07$	$4.63E-03$	$3.98E-3$	$2.24E-03$	$2.39E-03$	$1.99E-01$	$1.63E-01$
SumSquares	$1.28E-08$	$2.40E-08$	$2.36E-04$	$6.12E-04$	$1.38E-04$	$1.43E-04$	$1.37E-02$	$1.22E-02$
Booth	$3.66E-07$	$6.66E-07$	$3.53E-03$	$6.77E-03$	$1.39E-03$	$1.67E-03$	$5.32E-02$	$5.16E-02$
Levy-13	$2.01E-07$	$2.93E-07$	$2.98E-03$	$4.01E-03$	$5.99E-03$	$1.33E-02$	$1.10E-02$	$7.50E-02$

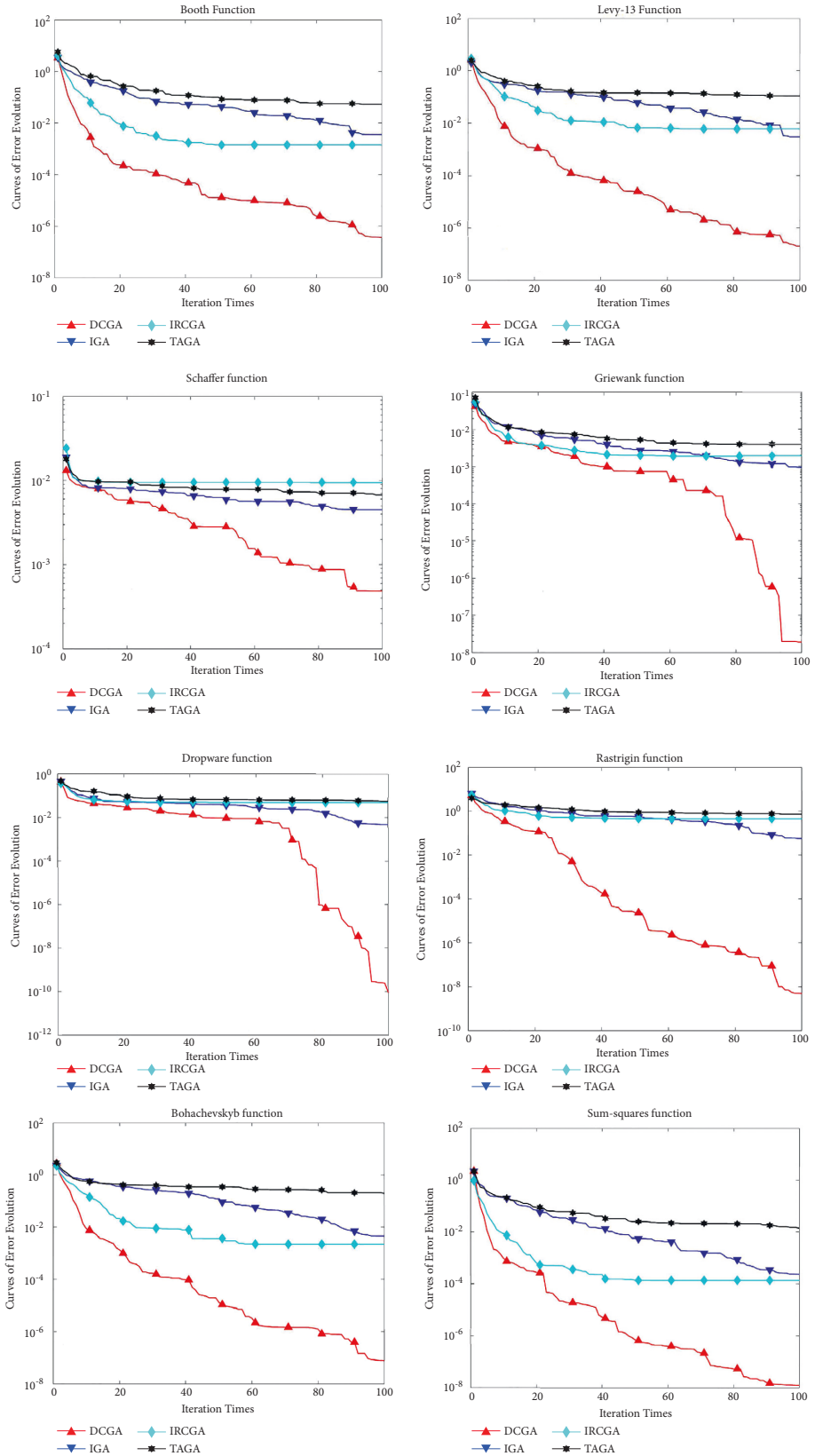

Figure 3: Curves of error evolution.

Figure 4: Template matching original image.

Table 2: Results of template matching.

Algorithms	Indicators			
	SSDA	GA	AGA	DCGA
Maximum time (s)	18.305	8.039	5.062	1.871
Minimum time (s)	15.506	7.681	3.832	0.817
Average time (s)	16.711	7.966	4.673	1.017
Correct numbers	49	45	46	48
Average matching accuracy	0.998	0.791	0.883	1

5. Conclusions

Aiming at the defects of slow convergence speed and premature convergence of traditional genetic algorithm, this paper proposes a multilayer mutation directed crossover genetic algorithm. On the one hand, the algorithm combines the crossover algorithm to improve the directional crossover operator and takes the current optimal solution of the population as the evolutionary goal to better reflect the evolution direction of the population; on the other hand, the algorithm introduces a multilayer adaptive operator to improve the accuracy of the degree of adaptation of different populations. The new algorithm and other comparison algorithms are compared in different dimensions for typical test functions and then compared in the model matching task. The two test results show that the optimization effect, convergence speed, and search ability of the algorithm have been greatly improved.

Data Availability

The test data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (62003122), Natural Science Foundation of Anhui Province of China (2008085UD03 and 1808085MF180), and the Fundamental Research Funds for the Central Universities of China (PA2020GDKC0016).

References

[1] B. Soucek, *Dynamic, Genetic, and Chaotic Programming: The Sixth-Generation*, John Wiley SonsInc, New York, NY, USA, 1992.

[2] D. I. Barnea and H. Silverman, "A class of algorithm for digital image registration," *IEEE Transactions on Computers*, vol. 21, pp. 179–186, 1972.

[3] G. K. Bayesteh, A. Alexander, and Sawetauk, "A survey of new techniques for image registration and mapping," *Applications of Digital Image Processing*, vol. 432, pp. 222–239, 1984.

[4] Z. Tian, R. Zhang, G. Sun, and D. Cheng, "Coordinated optimization of departure time domains of multiple trains at a station based on passenger satisfaction," *Journal of Control Science and Engineering*, vol. 2020, Article ID 1868724, 9 pages, 2020.

[5] L. Sun, X. Song, and T. Chen, "An improved convergence particle swarm optimization algorithm with random sampling of control parameters," *Journal of Control Science and Engineering*, vol. 2019, Article ID 7478498, 11 pages, 2019.

[6] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, MIT Press, Cambridge, MA, USA, 1992.

[7] Y. Liu, Y. Li, and C. Li, "Research on line loss calculation based on Genetic Algorithm Optimized BP neural network," *Computer Applications and Software*, vol. 36, no. 3, pp. 72–75, 2019.

[8] R. Liu and Y. Wang, "Research on TSP solution based on genetic algorithm," in *Proceedings of the 2019 IEEE/ACIS 18th International Conference on Computer and Information Science (ICIS)*, pp. 230–235, Beijing, China, June 2019.

[9] Y. Wang, "PCB drill path optimization by improved genetic algorithm," in *Proceedings of the 2019 5th International Conference on Control, Automation and Robotics (ICCAR)*, pp. 744–748, Beijing, China, April 2019.

[10] X. Yang and S. Chen, "Research on cloud resource allocation optimization algorithm under packet cluster mapping framework," *Computer Applications and Software*, vol. 36, no. 2, pp. 33–38, 2019.

[11] F. Zhao, R. Ma, and L. Wang, "A self-learning discrete jaya algorithm for multiobjective energy-efficient distributed No-idle flow-shop scheduling problem in heterogeneous factory system," *IEEE Transactions on Cybernetics*, pp. 1–12, 2021.

[12] F. Zhao, L. Zhao, L. Wang, and H. Song, "An ensemble discrete differential evolution for the distributed blocking flowshop scheduling with minimizing makespan criterion," *Expert Systems with Applications*, vol. 160, Article ID 113678, 2020.

[13] I. G. Tsoulos, A. Tzallas, and D. Tsalikakis, "PDoublePop: an implementation of parallel genetic algorithm for function optimization," *Computer Physics Communications*, vol. 209, pp. 183–189, 2016.

[14] Q. Yang, J. Jiang, Z. Qu, and Z. Guohong, "Application of logic operation to improve the performance of genetic algorithm," *Control and Decision*, vol. 4, pp. 510–512, 2001.
[15] S. Wang and Aorigele, "New strategy based on selection of mutation operator," *Computer Science*, vol. 41, no. 9, pp. 225–228, 2014.
[16] J. Li, "Hybrid particle swarm optimization algorithm for flexible job shop scheduling," *Computer Applications and Software*, vol. 32, no. 6, pp. 228–231, 2015.
[17] W. Liu, Y. Gong, W. Chen, Z. Liu, H. Wang, and J. Zhang, "Coordinated charging scheduling of electric vehicles: a mixed-variable differential evolution approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 12, pp. 5094–5109, 2020.
[18] S. Zhou, L. Xing, X. Zheng, N. Du, L. Wang, and Q. Zhang, "A self-adaptive differential evolution algorithm for scheduling a single batch-processing machine with arbitrary job sizes and release times," *IEEE Transactions on Cybernetics*, vol. 51, no. 3, pp. 1430–1442, 2021.
[19] F. Zhao, X. He, and L. Wang, "A two-stage cooperative evolutionary algorithm with problem-specific knowledge for energy-efficient scheduling of No-wait flow-shop problem," *IEEE Transactions on Cybernetics*, vol. 51, no. 11, pp. 5291–5303, 2021.
[20] X. Guan, *Improvement and Application of Genetic Algorithm under Real Number Coding*, Chongqing University, Chongqing, China, 2012.
[21] L. Jiao and L. Wang, "A novel genetic algorithm based on immunity," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 30, no. 5, pp. 552–561, 2000.
[22] Y. Song, F. Wang, and J. Lan, "Improved real genetic algorithm based on jumping gene operator," *Control and Decision*, vol. 35, no. 9, pp. 2277–2284, 2020.
[23] M. Srinivas and L. M. Patnaik, "Adaptive probabilities of crossover and mutation in genetic algorithms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 24, no. 4, pp. 656–667, 1994.
[24] Z. Wu, H. Shao, and X. Wu, "A new adaptive genetic algorithm its application in multimodal function optimization," *Control Theory and Application*, vol. 1, Article ID 127129, 1991.