Natural Computing (2021) 20:395–411
https://doi.org/10.1007/s11047-020-09830-2

A framework for designing of genetic operators automatically based on gene expression programming and differential evolution

Dazhi Jiang[1] · Zhihang Tian[1] · Zhihui He[1] · Geng Tu[1] · Ruixiang Huang[1]

Accepted: 27 November 2020 / Published online: 9 January 2021

Abstract
The design of genetic operators is absolutely one of the core work of evolutionary algorithms research. However, the essence of the evolutionary algorithms is that a lot of algorithm design is based on the manual result analysis, summarize, refine, feedback, and then, the algorithms are designed adaptively and correspondingly. This kind of design scheme needs artificial statistics and analysis of large amounts of data, which greatly increases the burden of the designers. To solve this problem, an evolutionary algorithm framework based on genetic operator automatic design is proposed in this paper. In the first step, Gene Expression Programming and Differential Evolution methods are combined together and used to design the genetic operators automatically and adaptively, this hybrid method can not only explore solutions in problem space for the problem solving as most classical evolutionary algorithms do, but also generate genetic operators automatically in operator space for the proper operators extraction and selection related to the evolutionary algorithms . In the second step, the designed operators are adopted into the typical evolutionary algorithms to verify the performance and the result shows that the new designed genetic operator is superior to or at least equivalent to some existing DE variants in a set of classical benchmark functions. More importantly, this paper is not aimed at designing high performance algorithms, but to provide a new perspective for algorithms designing, and to provide a reference scheme for the machine algorithms designing.

Keywords Evolutionary algorithm · Automatically designing · Self adaptive differential evolution algorithm · Gene expression programming

1 Introduction

In the 1990s, the genetic algorithm (Goldberg 1989; Preen and Smith 2019) has been into the booming period of rapid development, not only in theory research but also began to widely be used in applied research. The increase in the application field of the genetic algorithm is quickly, and what's more, the learning ability is also enhanced obviously. Application research has introduced a lot of new theories and methods and obtained rapid development, which has added new vitality to the research of genetic algorithm.

D. Whitey presents a cross crossover operator based on the field in 1991 and validated by the TSP problem (Arram and Ayob 2019). Seront and Bersini (1996) put forward the single operation of crossover operator by combining genetic algorithm with the single method, the generation of a candidate is made up of two parents and one other individual. After the comparison of results, the three individual's crossover operator has a better effect than point crossover and uniform crossover operator. There are quite a few domestic researchers continuously put forward new crossover operators of genetic algorithms, to optimize the genetic algorithm. In 1995, Kennedy and Eberhart presented a novel optimization technique, which named as

✉ Dazhi Jiang
dzjiang@stu.edu.cn

Zhihang Tian
18zxtian@stu.edu.cn

Zhihui He
Zhihuihe@aliyun.com

Geng Tu
19gtu@stu.edu.cn

Ruixiang Huang
15rxhuang@alumni.stu.edu.cn

[1] Department of Computer Science, Shantou University, Shantou 515063, China

PSO (Kennedy and Eberhart 1995; Ibrahim and Tawhid 2019). Similar to the traditional evolutionary algorithm (EA), PSO algorithm is a reliable and effective global optimization algorithm. It is also one of the most deeply studied and widely used algorithms. However, the essential contribution of the PSO algorithm lies in the effective design of the velocity and the position renewal equation based on the particle's position information, which are the current position, the best previous position, and the best position discovered. In 1997, Storn and Price presented a new kind of optimization technique, differential evolution (DE) (Storn and Price 1997; Antoniouk et al. 2019). The vector is generated by choosing two individuals from the parent to do the differential operation; Secondly, combine another selected individual and the differential vectors to generate the experimental individual. Then, the parent and the corresponding experimental individuals are crossed to generate new offspring individual. Similar to the PSO, DE and its variants are reliable and effective global optimization algorithms, which has been successfully applied in real world problems and benchmark problems.

There are numerous improved algorithms for SGA, PSO, and DE algorithm. But if we analyze them together, we will find that the genetic operators of these algorithms share some general schemes. First, the genetic operator is an arithmetic equation that is combined with objects/individuals, equation coefficients, etc. Second, based on the same objects or individuals, the arithmetic equation can have many variants by transforming operators, individual positions, and equation coefficients. Third, for many evolutionary algorithms, the most difference is essentially the difference in genetic operator equations.

Some special evolutionary algorithms can achieve excellent equation discovery and modeling capabilities. The typical methods are Genetic Programming (GP) (Koza 1992; Rodriguez-Coayahuitl et al. 2019), Gene Expression Programming (GEP) (Ferreira 2001; Xiao et al. 2019), and Multi Expression Programming (MEP) (Oltean and Grosan 2004). Or in other words, GP, GEP, and MEP can be used to generate arithmetic equations automatically for given input and output. Take the GEP method for example, for the arithmetic equation generating, the input of GEP could be the objects/individuals, equation coefficients, and arithmetic operators, and the output is the genetic operator for the given function optimization problems. If this idea can be realized, we think this paper has the following enlightenment significances for the research of evolutionary algorithms.

1. The algorithm can be designed by the machine automatically. For the traditional algorithm design, a typical design limitation is that: the intrinsic nature of algorithm design is attached to the manual design, and the design process is based on the repetitive and complex conception and experiment of the designers, tentative co-exist with the uncertainty. Thus, the efficiency of the algorithm design can't be guaranteed or expected. Based on machine design, although the design may not be as clever as manual design, it can give designers inspiration, perhaps some designed results obtained by machine are unimaginable in the aspect of human and beyond expectations.
2. The algorithm can be designed by the machine more adaptively. Generally, a lot of the research is based on the data statistical analysis to conclude the new difference algorithms. What's more, most of the existing learning strategies are not so adaptable for applicable to each problem function. So, it is important to generate different optimal operators on the different given problems. Let the machine generate the proper algorithm for the problems, which has great theoretical and practical value.

In this paper, with the powerful modeling ability of GEP, a framework for designing of genetic operators automatically is presented based the DE algorithm skeleton, and the main contributions are summarized as follows.

1. This paper put forward a basic concept and method for designing algorithms with the algorithm, which realizes the automatic design of the algorithm to a certain extent.
2. According to the characteristics of genetic operators in the DE algorithms, with the help of the GEP algorithm, a framework that can automatically generate genetic operators is constructed and validated on classical function optimization problems.
3. Comparing with the classical, effective, and hybrid SaDE algorithm (Qin et al. 2009), it is found that the framework of this algorithm can obtain competitive results.
4. To be honest, this paper does not aim at designing high-performance algorithms but designing a framework that can design algorithms automatically. We believe that this kind of research and exploration is still in its infancy, but it has potential research prospects. It is wondered that this automatic framework could be able to integrate with any high-performance algorithm to achieve stronger problem-solving ability.

The rest of this paper are organized as follows. The following section is the related work, and several classical and similar frameworks are described and discussed. Section 3 gives a process of automatic design of genetic operators based on GEP, and presents a framework of the automatic design of genetic operators. In Sect. 4, the experimental

verification of the algorithms is presented. In Sect. 5, conclusions and discussions were given.

2 Related work

Generally speaking, the design of automatic algorithms has attracted extensive attention to the evolutionary algorithm of academia. Furthermore, the main focus of automatic algorithm design is the selection of genetic operators and the influence of operator selection on the performance of current algorithms. For example, Zhang and Sanderson (2009) presented JADE, a new differential evolution (DE) algorithm, which could improves optimization performance by a new presented variation strategy. Brest and Maučec (2011) said that the selection of operators is critical for the DE performance, so they put forward two kinds of DE variants, which have two kinds of adaptive strategy selection technologies, namely, the probability match and the adaptive pursuit. Based on the typical multi-objective algorithm MOEA/D, Lin et al. (2017) presented an adaptive control strategy that could multiple DE strategies at different evolutionary stages adaptively. Based on the no free lunch theory, Mallipeddi et al. (2010) proposed an ensemble approach in which each mutation operator has its related population during the different stages of the problem-solving process. Jiang et al. (2014) and Jiang and Fan (2014) presented a novel evolutionary algorithm based on the automatic designing of genetic operators, which could find solutions in problem space and generate genetic operators automatically in operator space simultaneously. Based on the simple genetic algorithm, Diosan and Oltean (2009) presented a model to generate evolutionary algorithms by evolutionary means, which could find the optimal algorithm structure and variable parameter value to solve the problem more efficiently. SL-GEP was presented by Zhong et al. (2015). In the SL-GEP method, each chromosome in the population is composed of a main program and a set of automatically defined function (ADF) modules, and the defined functions are expressed by the gene expression of GEP. Then, in terms of mechanism, each ADF has functional units that can solve sub problems, which are combined into the main program to solve specific problems. Mahanipour and Nezamabadi-Pour (2019) presented a new gravitational search algorithm-based technique, called gravitational search programming (GSP) to create computer programs automatically. Nyathi and Pillay (2018) used genetic algorithm (GA) and grammatical evolution (GE) to design the GP-based classification algorithms automatically. In the experiments, the automated designed classifiers outperform manually designed classifiers. In view of the importance of mutation operator, Hong et al. (2018) used GP to generate operators automatically and the result shows that the proposed method outperforms existing human designed operators. Woodward and Swan (2012) presented a metal learning method to explore mutation operators in the constrained design space. Contreras-Bolton and Parada (2015) presented a combination mechanism for the crossover and mutation operators, which will improve the searching ability in the traveling salesman problem.

3 Designing of Genetic Operators Automatically based on GEP and DE

As mentioned before, the framework of the algorithm presented in this paper is based on GEP and DE algorithms, so this paper first reviews them. Then, based on GEP and DE, a detailed framework for designing genetic operators automatically is given. In addition, a program for verification is given to test the performance of new finding genetic operators.

3.1 GEP

As a linear expression programming method, GEP can be regarded as an extended algorithm which combines the advantages of GA and GP. Compared with GA, GEP utilized a new kind of encoding and decoding method to express and transform the individual, and essentially widely uses a hierarchical searching method to describe the problem. Furthermore, compared with the classical GP, the difference of GEP lies in that it uses a linearized genome to describe the individuals, and this genome can avoid combinatorial explosion in GP method.

Generally, the chromosomes of GEP consist of two parts, which are head (H) and tail (T) respectively. The gene in H is select from the operators set and the terminators set, whereas the gene in T can only selected from the terminators set (Ferreira 2001). Assuming h is the size of H, and t is the size of T, and all operations in operators set require at most n parameters, then t is a function which could be expressed:

$$t = h * (n - 1) + 1 \tag{1}$$

According to the rules mentioned above, a sample of chromosomes can be constructed as follows.

$$+Q - /b * aaQbaabaabbaaab$$

Like the phenotype of GP, the phenotype of GEP use the tree structure to describe the expression. According to the specific reading rules, the expression tree (ET) could be constructed. Thus, the above chromosome can be transformed into ET as the following (Fig. 1).

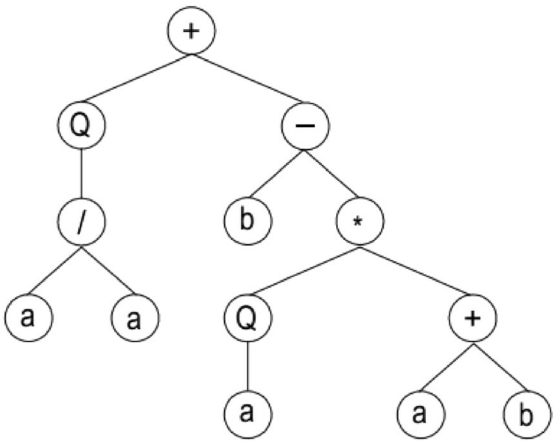

Fig. 1 A example of GEP expression tree

For this expression tree, we can travers it in an intermediate order, the equation could be obtained as follows.

$$Q(a/a) + (b - (Q(a) * (a + b)))$$

GEP genetic operators include mutation operators, transformation and insertion sequence elements and recombination operators. More details about the GEP genetic operators could obtained in the reference (Ferreira 2001).

3.2 DE

The idea of the DE originates from the genetic algorithm. As a robust and efficient evolutionary algorithm, DE has been widely used in the parallel computing, multi-objective optimization, constrained optimization, and numerous application areas. Assume N is the size of the population, D is the dimension and t is the generation, the population could be named as $X_i(t), i = 1, 2, \cdots, N$. The new vector $X_i(t+1)$ could be generated by:

$$X_i(t+1) = X_{r_1}(t) + F(X_{r_2}(t) - X_{r_3}(t)) \tag{2}$$

where $r_1, r_2, r_3 \in [0, N]$, and $r_1 \neq r_2 \neq r_3 \neq i$. F is the scale factor and $F \in [0, 1]$ usually.

The core part of DE is its differential strategy, which determines the optimization ability and efficiency. Early DE proposed many different differential strategies [9,10]. Some typical differential strategies are shown below.

$$X_1 + F * (X_2 + X_3 - X_4 - X_5) \tag{3}$$

$$X_b + F * (X_2 - X_3) \tag{4}$$

$$X_b + F * (X_2 + X_3 - X_4 - X_5) \tag{5}$$

$$X_1 + F_1 * (X_b - X_1) + F_2 * (X_2 - X_3) \tag{6}$$

$$X_i + F_1 * (X_1 - X_i) + F_2 * (X_2 - X_3) \tag{7}$$

$$X_i + F_1 * (X_b - X_i) + F_2 * (X_2 - X_3) \tag{8}$$

where X_1, X_2, X_3, X_4, X_5 represent random individuals, X_b is the best individual, X_i is the current individual, and F_1, F_2 are scaling factors, the first and fourth differential strategy are the most widely used currently, also one of the most successful differential strategy. As a matter of fact, in many different types of problems, this differential strategy (Liang et al. 2019) does not have the advantage to seek the optimal population or the optimal rate of convergence. If the problem is different, the corresponding difference strategy should also be changed, so as to be better adapted to the problem (Jiang et al. 2020). In this topic, the gene expression of evolutionary modeling method was adopted for different problems of optimal difference strategy. Modeling method by GEP evolution (Jiang and Fan 2014) can be automated design different expression that is suitable for different problems, and then use the expression in the differential evolution algorithm, in search of the best individual, as to solve the problem using approximate solution.

3.3 General Scheme of SGA, PSO and DE

If the individuals in the population can be regard as a particle, then the above genetic operators in SGA, PSO and DE can be regarded as the particle's position update equation(μ), which can be expressed as μ_{GA}, μ_{PSO} and μ_{DE}. Though the above analysis, we can found that the chromosome κ of GEP method can be mapped (the mapping equation defined as ζ) as an equation of the location update μ ($\mu = \zeta(\kappa)$), where the terminal set T is determined by the specific individuals in population, constants, random number, etc. The operator set O however, is determined by arithmetic operators, such as '+', '−', '*', etc.

According to the characteristics of chromosome, the longer the length ($l(\kappa)$) of chromosome, the more of genotypes and phenotypes, then the more particle's update equation can be mapped into. Suppose there are n chromosomes $(\kappa_1, \cdots, \kappa_n), n \to \infty$. Select one chromosome $\kappa_i, i \in \{1, \cdots, n\}$. When $l(\kappa_i) \geqslant L$, the following formula could be obtained.

$$\{\mu_{GA}, \mu_{PSO}, \mu_{DE}\} \subset \{\mu_1 = \zeta(\kappa_1), \cdots, \mu_n = \zeta(\kappa_n)\} \tag{9}$$

GA, PSO and DE's location update equations could be converted to the corresponding chromosomes according to the inverse operation of $\zeta(\zeta^{-1})$. Thus,

$$\kappa_{GA} = \zeta^{-1}(\mu_{GA}) \tag{10}$$

$$\kappa_{PSO} = \zeta^{-1}(\mu_{PSO}) \tag{11}$$

$$\kappa_{DE} = \zeta^{-1}(\mu_{DE}) \tag{12}$$

Assuming all of the genes in chromosomes of κ_{GA}, κ_{PSO}

and κ_{DE} are effective genes (the definition of effective gene see (Jiang et al. 2006)), then,

$$L = \min(l(\kappa_{GA}), l(\kappa_{PSO}), l(\kappa_{DE})) \tag{13}$$

According to the analysis above, we can found that one obvious advantage of chromosome κ is its strong representation capability, and μ_{GA}, μ_{PSO} and μ_{DE} can be seen as s special case of κ. Suppose a chromosome with length L (Take GEP method for example, the traditional chromosome of GEP are constructed by the head unit and tail unit usually. The genes in head are selected from T and O randomly, and genes in tail are only selected from set O. The head length is h, the tail length is t, the total length $L = h + t = h + h(n-1) + 1 = hn + 1$, the size of set T is α and O is β), the number of the location update equations ($\mathbb{N}$) can be expressed as follows.

$$\mathbb{N} = \prod_{j=1}^{h} (\alpha + \beta) \prod_{k=h+1}^{hn+1} \beta \tag{14}$$

Suppose h=10, n=2, α=5 and β=3, then, $\mathbb{N}$=190210142896128.

Since chromosomes have such representation ability in automatic programming methods, can automatic programming methods automatically construct genetic operators of evolutionary algorithms and generate operators in the process of solving problems instead of pre-defined ones? Generally, evolutionary algorithms have the characteristics of self-organization, self-adaptive and self-adjusting. However, the genetic operator of an evolutionary algorithm is always determined in advance. The algorithm composed of predetermined operators may be particularly effective in some problems, but it does not necessarily show the same degree of superiority in other problems. This phenomenon could be explained by NO Free Lunch (NFL) theory (Wolpert and Macready 1997). So can we make a hypothesis, if one algorithm's genetic operators also have self-organizing, self-adapting, and self-regulating capacities, which means the genetic operator could automatically adjust to the problems in the problem-solving process, does the NFL theory could be broke seemly? Because the chromosome in automatic programming methods has a strong ability to express the equations, if the chromosome is mapped into the particle's update equation, then the update equation may be incogitable, or cannot be designed manually by people. It can be said that using the automatic programming method to evolve genetic operators can realize the automation of evolutionary algorithms to some extent. This is a quite bold and novel concept. Of course, GP, GE, GEP and MEP methods all can be used to construct the genetic operators. In this paper, the GEP method is selected as the algorithm carrier of automatic genetic operator generation.

3.4 The framework for designing of genetic operators automatically based on GEP (DGOA)

In this paper, the effectiveness of the automatic framework is verified by using a set of single objective optimization problems first. To achieve this goal, the framework is generally divided into two modules, the first one is the genetic operators designing module, and the other is the function optimization module.

The two modules have different responsibilities. In the genetic operators designing module, the duty of the GEP based method is to find the proper genetic operators automatically. Meanwhile, in the function optimization module, the operators, generated by the genetic operators designing module, are obtained to manipulate the individuals for function optimization, in order to find the global optimal solution of the problem.

Obviously, in the automatic design of genetic operators, the genetic operators are not defined by the designer before solving the problem. In fact, genetic operators are searched and designed in the process of solving problems. For this consideration, two models run at the same time and interact with each other to realize genetic operators' discovery and problem solutions discovery (see Fig. 2).

The interaction between the two modules can be described as follows: In the genetic operators designing module, GEP method is used to a variety of genetic operators for differential evolution algorithm, then apply them into the DE algorithm, the DE adopted to the evaluate the efficiency of genetic operator and then feedback to GEP algorithm.

First initializing the genetic operators, and then using the selection operator, mutation operator, the string of operator, and recombination operator methods to generate a new genetic operator, and apply the generated genetic operator to DE algorithm, calculate the optimal value of the corresponding problem. If the optimal value does not meet the optimal value desired, then continue the process of selecting the operator, mutation operator, the string of operator, and recombination operator to adaptive a new genetic operator, until the optimal value of calculating meet the expected, then break the circulation. So the method of GEP evolution automation design modeling is suitable for generating differential expression of different problem (Chen et al. 2017; Song et al. 2020; Jiang et al. 2020).

Because the genetic operators produced by genetic operators designing module are unpredictable, the differential operators may be good or bad. For these unforeseen operators, some operators need to be avoided, such as the follows:

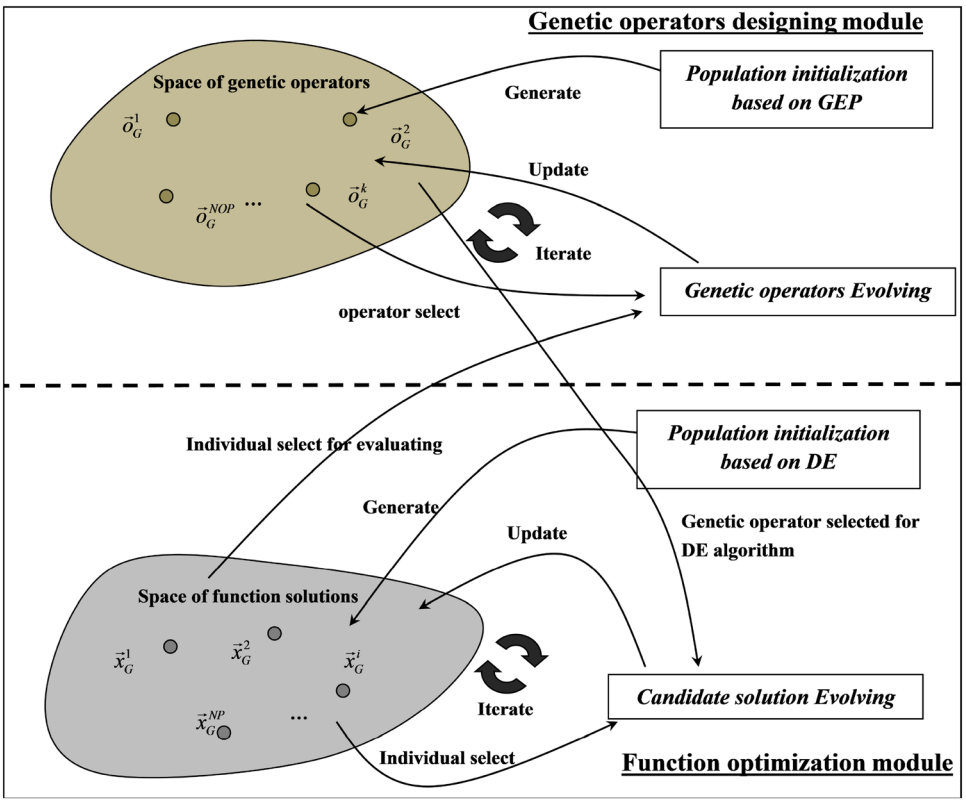

Fig. 2 The general diagram of DGOA

$$X_i - F * X_j \tag{15}$$

$$X_i - X_i \tag{16}$$

In Formula 16, it can be seen that the operators only contain two individuals, and the diversity of two individuals is smaller than that of three individuals. Therefore, it is necessary for the DGOA algorithm avoid generating such operators (Fig. 3).

In the formula 16, we can clearly know that fitness for this operates will be the constant 0. No matter how many individual values participate in the mutation, the application of this formula leads to the DE algorithm cannot produce new changes, and thus falls into local search.

To solve the problems mentioned above, a detection method is presented in this paper to effectively avoid the generation of invalid operators: according to the dimension D of the problem, M different test vectors $T_i, i \in [1, M]$ are generated randomly. By substituting M test vectors into genetic operator G, M new individuals $T_i^{'}$ can be obtained, and then M individuals can be compared whether they are invariant or not. If it is invariable, the difference strategy is invalid and cannot cause changes, which means it is not proper applied to DE algorithm; on the contrary, it is effective.

3.5 Program for verification

In order to verify the performance of DGOA, two programs for verification are designed. For the first program, it is very similar to the classical DE algorithm, but the only difference is the genetic operator. For the second program, we choose the SaDE algorithm (Qin et al. 2009). SaDE is a classical and effective hybrid algorithm for complex function optimization. Two genetic operators generated by DGOA algorithm are selected and add them to SaDE to further verify the performance. DE algorithm is familiar and classical, so in this paper, we focus on the second program.

As has been said in the previous section, GEP model are used to automatically evolve genetic operators. This chapter will talk about genetic operator combination of GEP evolution, which can greatly reduce the DE algorithm trapped in local optimal solution and can't walk out of trouble. The main steps of hybrid evolutionary algorithm based on DE algorithm are common, with not big difference, in the process of DE algorithm, ordinary DE is to use a genetic operator to update the population iterative, but in a hybrid evolutionary algorithm using two genetic operators.

Since two candidate learning strategies have been selected, it is assumed that one strategy can be selected for each individual in the current population, and the

probability of applying one learning strategy is $p1$, and the probability of applying the other is $p2$, where $p2 = 1 - p1$. In the initialization phase, $p1 = p2 = 0.5$. Then, in the process of solving the problem, $p1$ and $p2$ are not fixed, but dynamically adjusted. After evaluation of all newly generated test candidates, record the test candidates generated by strategy one and successfully enter the next generation. The total number of recorded candidates is marked as $ns1$, and the success test candidates generated strategy two is $ns2$. What's more, we also record the failure of the two strategies. The number of discarded test candidates generated by strategy one is recorded as $nf1$, and the number of discarded test candidates generated by strategy two is recorded as $nf2$. The probability of $p1$ and $p2$ are updated as follows:

$$p1 = \frac{ns1 * (ns2 + nf2)}{ns2 * (ns1 + nf1) + ns1 * (ns2 + nf2)} \tag{17}$$

$$p2 = 1 - p1 \tag{18}$$

At last we adapted the generation using two candidates learning strategies applied according to the probability of $p1$ and $p2$ to conduct the comprehensive experiments, and then comparing the performance with SaDE.

The framework of hybrid evolutionary algorithm based on DE

1. **Begin**
2. **Input**: function optimization problem
3. Initializes the GEP related parameters, initializes the best expression *bestExpr*
4. **while** *iter* <*MaxGen* **do**
5. Gets individual in the GEP population (*kExprMp*) in the **Genetic operators designing module**and uses it to process the corresponding function in the **Function optimization module**
6. Returns the fitness value *ret*.
7. **If** *ret* <*bestValue* **then**
8. *bestValue*=*ret*, save the *kExprMp* to the next iteration
9. **End if**
10. Update the global best solution
11. Genetic operators' evolution by the traditional GEP operators
12. *iter* = *iter* + 1
13. **End while**
14. **Output** the best expression (bestExpr) for the test functions, and choose two different strategies to replace the genetic operation of classical DE for the following evolution.
15. Loading control parameters for DE: F, CR, NP, G=0
16. Initialization the probability of two difference strategies: $p1$=$p2$=0.5, $ns1$=$ns2$=0, $nf1$=$nf2$=0
17. **While** stopping criterion is not satisfied
18. For the population NP, randomly generate a vector of size NP, $v_{jth} = rand_j\,[0, 1]$ for each element
19. Select one of the difference strategies according to the probability $p1$, $p2$
20. **If** $(v_{jth} < p1)$, then choose strategy one, k=1
21. **Else** select the strategy two and apply it to the DE algorithm, k=2
22. **End If**
23. $\overrightarrow{u}^{r}_{j,G}$ =fitness evaluation $\left(\overrightarrow{x}^{r_1}_{j,G}, \overrightarrow{x}^{r_2}_{j,G}, \overrightarrow{x}^{r_3}_{j,G}, \overrightarrow{x}^{best}_{j,G}, \overrightarrow{o}^{k}_{G}\right)$
24. Calculate the fitness of individual species $f\left(\overrightarrow{u}^{r}_{j,G}\right)$
25. **If** $f\left(\overrightarrow{u}^{r}_{j,G}\right) \leqslant f\left(\overrightarrow{x}^{r}_{j,G}\right)$, then $\overrightarrow{x}^{r}_{j,G+1} = \overrightarrow{u}^{r}_{j,G}, f\left(\overrightarrow{x}^{r}_{j,G+1}\right) = f\left(\overrightarrow{u}^{r}_{j,G}\right)$, and $nsk = nsk + 1$
26. **If** $f\left(\overrightarrow{u}^{r}_{j,G}\right) < f\left(\overrightarrow{u}^{r}_{best,G}\right)$, then $\overrightarrow{x}^{r}_{best,G} = \overrightarrow{u}^{r}_{j,G}, f\left(\overrightarrow{x}^{r}_{best,G+1}\right) = f\left(\overrightarrow{u}^{r}_{j,G}\right)$
27. **End if**
28. **Else**
29. $nfk = nfk + 1$
30. **End if**
31. $G = G + 1$
32. **End while**

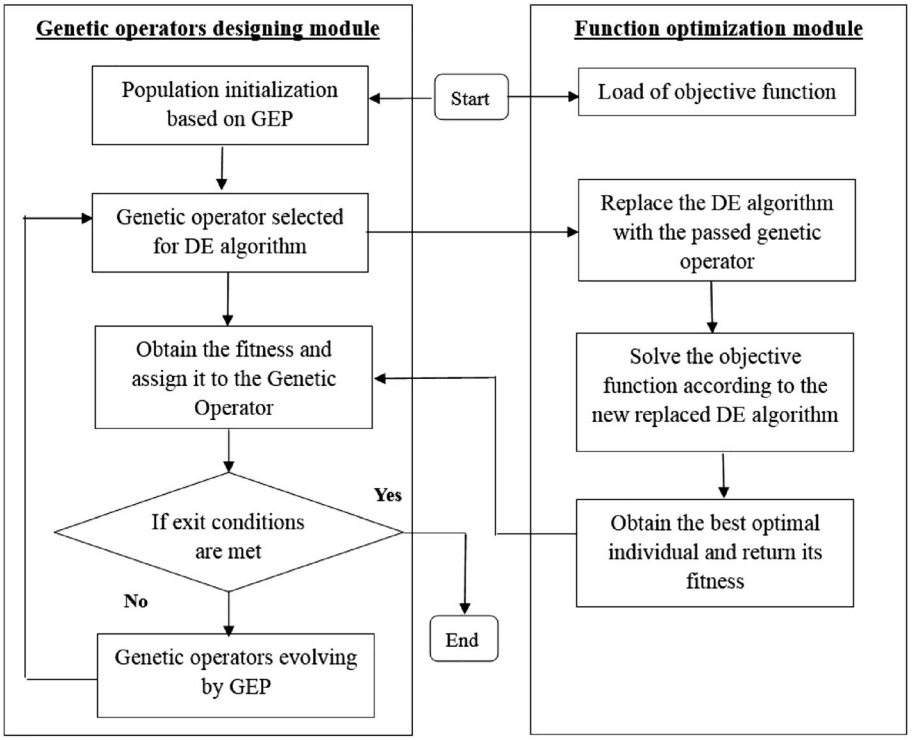

Fig. 3 The framework of DGOA

4 Experimental Verification

4.1 Problem Description

Function minimization problem is a classical differential evolution problem: given a set of functions and its scope, and calculate the minimum of the function within the scope, the following Table 1 lists the function of eight:

4.2 The operators generated by DGOA

Table 2 shows the parameter setting for the DGOA. For genetic operators designing module, $T = \{X_i, X_j, X_k, F\}$ and $F = \{+, -, *\}$. It can be seen that in order to simplify the program design of this topic, only three individuals and one scaling factor are selected in this paper. In addition, the genetic operators generated by GEP can only contain three operators in F. In fact, the individual possibility of the difference operators composed of T and F is enough to satisfy most of the problem-solving needs. Of course, it could be imagined that there will be more and more complex operators with the different selection of T and F. Other parameters of DE algorithm are set according to classical DE algorithm.

In this experiment, DGOA automatically evolved difference operators for the different function optimization problem and the results of the optimum operator obtained in the different generation is showed as follows (Table 3).

Due to the time limits of the DGOA, the max generation program is set as 120, so when the iteration to 120 even if haven't reached in the required accuracy and stop the iteration. It can be known that only the F7 did not get the expectations after max generation from the experiment, F1–F6 and F8 problems, under the less number of iterations approximation expectation to the required accuracy value of DE algorithm is close to expectations, then jumping out of the DGOA cycle. So, it can be identified that DGOA is close to the performance of DE in the ability of the minimum result finding, which means that the genetic operators generated by the DGOA are effective and reasonable (Fig. 4). The most excellent operators generated in the different problems are shown as follows.

$$\text{Strategy1} : X_i + X_i * X_j * F * F * X_j + X_i * X_i \tag{19}$$

Table 1 The test example of function to minimize

No.	Function	R	Min
F1	$\sum_{i=1}^{D} x_i^2$	[− 100,100]	0
F2	$\sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	[− 10,10]	0
F3	$\sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_i\right)^2$	[− 100,100]	0
F4	$\sum_{i=1}^{D} \left[100\left(x_{i+1} - x_i^2\right)^2 + (1 - x_i)^2\right]$	[− 30,30]	0
F5	$\sum_{i=1}^{D} floor(x_i + 0.5)^2$	[− 100,100]	0
F6	$\sum_{i=1}^{D} i * x_i^4$	[− 1.28,1.28]	0
F7	$-\sum_{i=1}^{D} \left(x_i * \sin\sqrt{\lvert x_i \rvert}\right)$	[− 500,500]	D=30 − 12569.5
F8	$\sum_{i=1}^{D} \left[x_i^2 - 10\cos(2\prod x_i) + 10\right]$	[− 5.12,5.12]	0

Table 2 The parameters setting in the proposed DGOA

Methods	Parameter	Parameter description	Value
Function optimization module	D	Dimension of the problems	30
	S	Genetic operator used	$X_1 + F * (X_2 - X_3)$
	DE_{Gen}	The maximum generation	500
Genetic operators designing module	F	Function set	+, −, *
	T	Terminal set	$\{X_i, X_j, X_k, F\}$
	P_{size}	Population size	50
	GEP_{Gen}	The maximum generation	120
	$R_{Mutation}$	The probability of mutation	0.01
	$R_{One-point}$	The probability of one-point recombination	0.3
	$R_{Two-point}$	The probability of two-point recombination	0.3
	R_{gen}	The probability of gene recombination	0.1
	R_{IS}	The probability of transposition	0.1
	R_{RIS}	The probability of root transposition	0.1
	R_{gen}	The probability of gene transposition	0.1

$$\text{Strategy2} : F * \left(X_i + X_j + X_k\right) \tag{20}$$

$$\text{Strategy3} : X_i * X_k * X_k \tag{21}$$

$$\text{Strategy4} : X_j + X_j * X_j * X_j + F * X_j * X_j \tag{22}$$

$$\text{Strategy5} : X_i + X_i * X_i * X_j + F * X_i + X_i + X_i * X_i \tag{23}$$

$$\text{Strategy6} : F * X_j * X_j - X_j * X_j * X_k \tag{24}$$

$$\text{Strategy7} : X_j + F * X_i - F * X_k + F \tag{25}$$

$$\text{Strategy8} : F * X_k * X_k * X_k \tag{26}$$

4.3 DE based on the outcome of GEP

This is the structure of general genetic operator of DE algorithm. We have selected eight candidate learning strategies, which are eight better performed genetic operators that have generated from the novel evolutionary algorithm in the evolution of GEP. Eight operators respectively apply in eight questions function. Comparing the performance of all genetic operators in these eight functions, several genetic operators (two operators are enough in our research) are selected as candidate operators, which will be used in hybrid evolution algorithm.

Table 3 The optimum operator obtained in the different generation for different function optimization problem

No.	Generation	Optimum operators obtained	Optimum result obtained
F1	15	$X_i + F + F - X_k + X_j + F - X_j + X_j + X_i + F * X_k$	13156.758642708399
	30	$F - X_j + X_j + X_i + F * F * X_k + X_j * X_k * X_k + X_i$	5105.1655715281
	45	$F - X_i + F - X_i * F * X_j - X_j + X_i + F - X_j * X_j$	3685.9694916068
	60	$F - X_j - F + X_k + F + F + F - X_i + F - X_k$	940.6043981833
	75	$F - X_j - F + X_k + F + F + F - X_i + F - X_k$	699.3918522054
	90	$X_i + X_j * X_i * X_k - X_j + X_k * X_i + X_j$	0.0020499087
	105	$X_i + X_j * X_i * X_k - X_j + X_k * X_i + X_j$	0.0001621843
	120	$X_i + X_i * X_j * F * F * X_j + X_i * X_i$	0.0000070843
F2	15	$X_k + X_k + F - F - X_k + X_k + X_i + X_k$	0.01127982234437
	30	$X_j + X_k + X_i + X_i + X_k + X_k$	0.00386592893661
	45	$X_i + X_i + X_k * X_j + X_k * X_j + X_i - X_k - X_k$	0.00294884948572
	60	$X_i * X_k - X_i * X_k + X_k + X_j + X_k + X_k + X_k$	0.0003008209
	75	$X_k + X_i - X_k * X_j + X_i + X_k + X_i + X_k + X_k$	0.0002578230
	90	$X_k + X_i - X_k * X_j + X_i + X_k + X_i + X_k + X_k$	0.0000780300
	105	$F - X_i * X_k * X_j + X_k - X_i + X_i + X_i + X_k + X_j - F - F$	0.0000137329
	120	$X_i * F + X_k * F + X_j * F$	0.0000120254
F3	15	$X_i * X_j * X_i - X_k + F + X_j - X_k + F - F * X_i + X_i * X_j + X_i$	13709.488171389434
	30	$X_k + X_i + X_k * F + X_k * X_i - X_i * X_k + X_i * X_j + X_k$	10939.851247400038
	45	$X_j * F + F + F + X_i * X_k * X_i + X_j - X_k + F - F * X_i$	9392.271507068459
	60	$X_k - X_k * X_j - F + X_j + X_i * X_k * X_i + X_k * X_k * F$	8051.1615334301
	75	$X_j - F * X_k * X_i + X_k - X_k * F - X_k * X_i + X_k - X_j + F * X_k * X_j$	7181.3836076108
	90	$X_i - X_k * X_i * X_i + X_i * X_k + X_i + X_j * F - F * X_j$	6099.0539030968
	105	$X_j * X_j * X_i + X_i * X_k - X_i * X_i + X_i * X_i * X_i * F$	4654.6998505466
	120	$X_i * X_k * X_k + F - X_j + X_i + X_j - X_i - F$	0.0000000000
F4	15	$X_j * F + X_k - F + F * X_i + X_k - X_i + X_i$	995426.7689860109
	30	$X_k * X_i + X_j + X_k + X_i - X_k + X_i * X_i + X_j + X_j$	538690.2178677756
	45	$X_k + X_j + X_k + X_i * X_k * X_j + X_k - F + X_j * X_i$	477980.1780576324
	60	$X_k * X_j * X_j + F - F + F * X_k + X_k$	99611.0796244393
	75	$X_k * X_j * X_j + F - F + F * X_k + X_k$	1.2276266925
	90	$X_k - X_j - X_k + X_j + X_k * X_i + X_i * X_k + X_i * X_i$	0.0010336278
	105	$X_i * X_k * F + X_j - X_j + X_k * X_j * X_j$	0.0000678148
	120	$X_j + X_j * X_j * X_j + X_j * X_j * F$	0.0000012738
F5	15	$X_k - X_k - X_k + X_i + X_j - X_k + X_j + X_i + F - X_j + F - X_k$	15774.0
	30	$X_i + F + F - X_j + F - X_k + F + X_k * X_i - X_i$	8383.0
	45	$F + X_k * X_i - X_i + X_j - X_j + X_k - X_k - X_j$	b6873.0
	60	$X_j - X_j + X_j - X_j + X_k - F - X_j$	141.0
	75	$X_k - X_k - X_k + X_i + F + X_j - X_j$	106.0
	90	$F - X_i - X_i - X_j + X_i + X_j + X_k - X_k - X_j$	12.0000000000
	105	$F - X_i - X_i - X_j + X_i + X_j + X_k - X_k - X_j$	8.0
	120	$X_i + X_i * X_j * X_i + F * X_i - F + F + X_i + X_i * X_i$	0.0000000000

Table 3 (continued)

No.	Generation	Optimum operators obtained	Optimum result obtained
F6	15	$X_k * X_i + X_j - X_i - X_j + X_j - X_i + X_j$	1.3509759354263E8
	30	$F - X_j + X_j - F - X_j + X_k * X_i - X_j * X_k * X_j$	4.1799822283401E7
	45	$F - X_j * X_k * X_j + X_k * X_j - X_j * X_i * X_k + X_j + X_k - F - X_j$	16890408.61172307
	60	$F - X_j * X_k * X_j + X_k * X_j - X_j * X_i * X_k + X_j + X_k - F - X_j$	223006.7602112475
	75	$F * X_i * X_i + X_i * X_k - X_j * X_i + X_i$	0.0000000002
	90	$F * X_i * X_i + X_i * X_k - X_j * X_i + X_i$	1.990165719822E-10
	105	$X_i + X_j - X_i - X_j + X_j - X_j * X_j * X_k + F * X_j * X_j - X_j$	7.168213463290E-22
	120	$X_i + X_j - X_i - X_j + X_j - X_j * X_j * X_k + F * X_j * X_j - X_j$	0.0000000000
F7	15	$X_i * X_k - X_k + F + X_i * X_j + X_j * X_j - F + X_j - X_i$	−3398.7645476988
	30	$X_j + X_i - F + X_i * X_k - X_k + F + X_i + F$	−3557.1707681105
	45	$X_i + X_j + F * X_j + F + X_j - X_k + X_j - X_k + F$	−3613.4926845013
	60	$F * X_k - X_i + X_i + F + X_j - F$	−3628.0611730137
	75	$F + X_k - F + X_j * F * F - X_i + X_i + F$	−3770.3089248318
	90	$F + X_k - F + X_j * F * F - X_i + X_i + F$	−3848.9293403834
	105	$X_j + X_j - X_i * X_j * X_i - F + X_k - X_i + X_k + F$	−3997.7629197805
	120	$X_j + F * X_i - F * X_k + F$	−4568.8421573061
F8	15	$X_j + F + X_i + X_j + F * F + F * X_i + X_k$	12879.494515023642
	30	$X_i + X_k + F * F - X_i * X_j + X_i$	4936.5756210779
	45	$F + X_j * X_i * X_j - X_i + F + X_k - X_j + X_k + X_j + F * X_i$	4841.7033589451
	60	$X_j + X_j + X_i + X_i * X_j - X_i - F + F + X_j$	567.6872962055
	75	$X_i * X_k * X_i + F - F + X_i * X_j * X_k + X_k - X_i + X_i$	1.0510238405
	90	$X_i * X_k * X_i + F - F + X_i * X_j * X_k + X_k - X_i + X_i$	0.0412137754
	105	$X_k - X_i * X_k + X_k - X_i + X_i$	0.0019712209
	120	$F * X_k * X_k * X_k + X_i + F - F + X_j + F - X_i - F - X_j$	0.0000000000

After finishing the experiment, eight appropriate difference strategy were selected in the experiment results. The following table is the mean results of each function based on DE algorithm, using the eight genetic operators selected, where Max_FEs=100000 and 25 independent runs were executed (Tables 4, 5 6, 7).

From the line chart above, it is obvious that the difference strategy8 converge much earlier to expectation value than classical differential strategy DE algorithm. So in comparison, for function 8, differential strategy which automatic designed based on GEP performed much better.

From the experimental results, it can be known that the seventh function optimization is not exactly reached the expected value, but the performance is almost the same as classic DE difference strategy. And the rest of the functions in the best of circumstances are reached the expected value. In with other functions, however, the median is completely meet the expectations of each function, even the worst case, and only one value didn't meet the expectations, it shows that the DE evolutionary algorithm based on GEP for minimum problem of the function is effective.

Compared with the used of classic genetic operators in DE algorithm, the performance of difference strategy that automatic designed based on GEP is higher, the better.

4.4 The comparative experiment with SaDE

Two genetic operators generated by DGOA algorithm are selected and add them to SaDE to further verify the performance. Through this experiment (Fig. 5), we want to verify that although DGOA is oriented to the genetic operators generated by independent and specific problems, these operators have certain wide adaptability. Here, we present an Adaptive and hybrid DE (AdhDE) based on the selected genetic operators designed by the DGOA, and the selected operators are $X_k * F * F - X_k + X_i + X_k + X_k - X_k - X_j$ and $X_k + F * X_k - X_j + X_i * F + F * F$. In the SaDE, the selected operators are $X_i + F * (X_j - X_k)$ and $X_i + F * (X_b - X_i) + F * (X_2 - X_3)$. In the following figures, AdhDE stands for the improved method of Automatic Designed Hybrid differential strategy that based on DE.

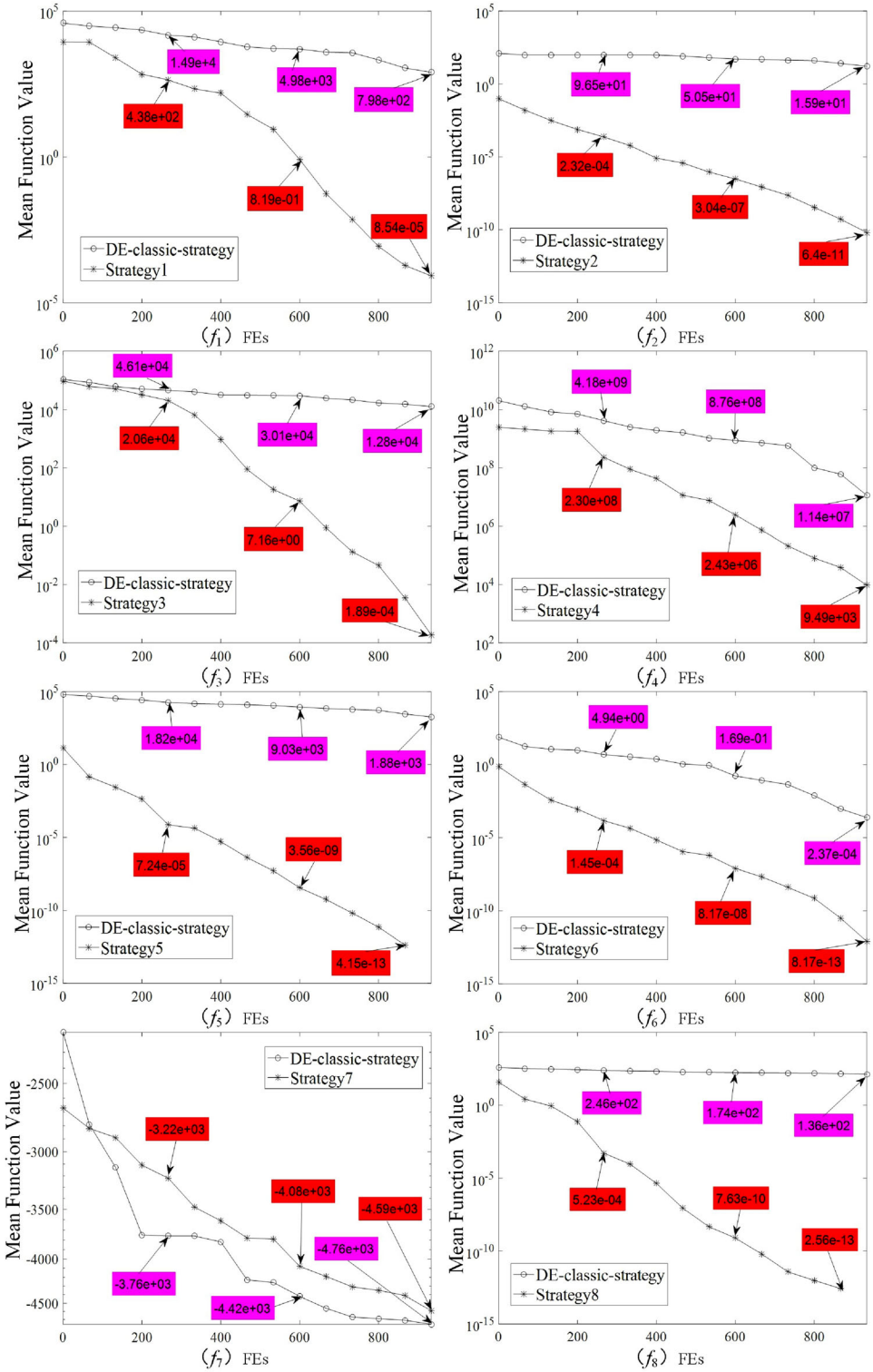

Fig. 4 The convergence speed between Classical DE and the operators generated by the DGOA

The line chart below is about the mean value obtained by the AdhDE and SaDE, Max_FEs=100000 and 25 independent runs were executed.

By observing the experimental results, it can be found that most of the mixed operator (DE) of automatically generated by the GEP algorithm can get the better minimum value than SaDE, and can be closer to the optimum value. From the line chart, we can find that while the function is approaching to the minimum, most of the

Table 4 For function 2

FEs		Strategy1	Strategy2	Strategy3	Strategy4	Strategy5	Strategy6	Strategy7	Strategy8	DE−rand1
1000	Best	1.01E−05	1.60E−09	2.99E−03	3.98E+00	5.13E+01	1.36E−08	5.49E−01	8.00E−10	9.23E+00
	Median	1.45E−04	1.60E−09	7.93E−03	9.23E+00	6.68E+01	9.95E−07	5.57E−01	1.99E−07	1.34E+01
	Worst	6.61E−02	1.60E−09	1.56E+00	1.78E+01	7.78E+01	1.30E−04	5.65E−01	1.58E−04	3.64E+01
	Mean	3.29E−03	6.40E−11	1.32E−01	9.86E+00	6.59E+01	1.57E−05	5.56E−01	1.41E−05	1.59E+01
	Std	1.59E−04	9.44E−20	1.15E−01	1.17E+01	2.97E+01	1.16E−09	1.69E−05	1.27E−09	3.41E+01
10000	Best	3.00E−10	0.00E+00	2.08E−03	3.83 E+00	1.14 E+00	0.00E+00	5.32E−01	0.00E+00	3.99E−07
	Median	2.20E−09	0.00E+00	2.55E−03	8.19E+00	4.07E+00	0.00E+00	5.44E−01	0.00E+00	1.09E−06
	Worst	5.10E−09	0.00E+00	2.75E−03	1.50E+01	9.67E+00	1.00E−10	5.52E−01	1.00E−10	1.11E−05
	Mean	2.08E−09	0.00E+00	2.54E−03	8.86E+00	4.23E+00	1.60E−11	5.43E−01	1.60E−11	2.02E−06
	Std	2.31E−18	0.00E+00	1.81E−08	6.60E+00	3.83 E+00	1.30E−21	1.18E−05	1.30E−21	5.48E−12
100000	Best	0.00E+00	0.00E+00	4.28E−05	1.00E−10	1.25E−01	0.00E+00	8.50E−03	0.00E+00	2.20E−09
	Median	0.00E+00	0.00E+00	5.19E−05	1.00E−10	1.34E−01	0.00E+00	1.01E−02	0.00E+00	2.30E−09
	Worst	1.00E−10	0.00E+00	2.65E−03	2.09E+01	1.39E−01	1.00E−10	5.41E−01	1.00E−10	2.40E−09
	Mean	4.00E−11	0.00E+00	2.57E−03	8.70E+00	1.33E−01	3.20E−11	5.45E−01	2.80E−11	2.33E−09
	Std	2.49E−21	0.00E+00	1.82E−06	3.98E+00	5.47E−03	2.18E−21	8.44E−05	1.99E−21	1.67E−18

Table 5 For function 5

FEs		Strategy1	Strategy2	Strategy3	Strategy4	Strategy5	Strategy6	Strategy7	Strategy8	DE−rand1
1000	Best	0.00E+00	0.00E+00	0.00E+00	7.93E+02	0.00E+00	6.85E+03	0.00E+00	6.04E+03	8.42E+02
	Median	0.00E+00	0.00E+00	0.00E+00	3.71E+03	0.00E+00	1.03E+04	0.00E+00	1.03E+04	1.78E+03
	Worst	0.00E+00	0.00E+00	0.00E+00	7.00E+03	0.00E+00	1.44E+04	0.00E+00	1.42E+04	3.77E+03
	Mean	0.00E+00	0.00E+00	0.00E+00	3.33E+03	0.00E+00	1.07E+04	0.00E+00	1.03E+04	1.88E+03
	Std	0.00E+00	0.00E+00	0.00E+00	2.88E+06	0.00E+00	6.50E+03	0.00E+00	4.93E+03	5.52E+03
10000	Best	0.00E+00	0.00E+00	0.00E+00	1.35E+03	0.00E+00	1.64E+03	0.00E+00	2.72E+03	0.00E+00
	Median	0.00E+00	0.00E+00	0.00E+00	3.71E+03	0.00E+00	6.56E+03	0.00E+00	6.31E+03	1.00E+00
	Worst	0.00E+00	0.00E+00	0.00E+00	8.43E+03	0.00E+00	9.50E+03	0.00E+00	1.18E+04	4.00E+00
	Mean	0.00E+00	0.00E+00	0.00E+00	4.01E+03	0.00E+00	6.07E+03	0.00E+00	6.49E+03	9.60E−01
	Std	0.00E+00	0.00E+00	0.00E+00	6.72E+03	0.00E+00	1.28E+03	0.00E+00	1.46E+03	1.13E+00
100000	Best	0.00E+00	0.00E+00	0.00E+00	5.76E+03	0.00E+00	5.00E+02	0.00E+00	5.09E+02	0.00E+00
	Median	0.00E+00	0.00E+00	0.00E+00	2.47E+03	0.00E+00	3.07E+03	0.00E+00	3.88E+03	0.00E+00
	Worst	0.00E+00	0.00E+00	0.00E+00	4.18E+03	0.00E+00	6.29E+03	0.00E+00	6.95E+03	3.00E+00
	Mean	0.00E+00	0.00E+00	0.00E+00	2.29E+03	0.00E+00	3.06E+03	0.00E+00	3.89E+03	4.80E−01
	Std	0.00E+00	0.00E+00	0.00E+00	1.98E+03	0.00E+00	4.33E+03	0.00E+00	4.99E+03	5.61E−01

function that used hybrid operator algorithm has more quickly convergence speed than traditional SaDE.

It shows that the hybrid evolutionary algorithm based on DE for minimum problem of the function is effective. Even the occasional results do not meet expectations, but the hybrid evolutionary algorithm based on DE overall is stable, reasonable and effective.

5 Conclusion

The content of this topic in research is to explore the GEP evolution modeling, and the performance of the generated genetic operators under the applications of DE algorithm. Then comparing the performance with SaDE algorithm, that is to say, comparing the performance of hybrid operator which generated by GEP evolution with classic SaDE operator. It can be found that our evolution algorithm of hybrid genetic operator is better on the convergence of

Table 6 For function 8

FEs		Strategy1	Strategy2	Strategy3	Strategy4	Strategy5	Strategy6	Strategy7	Strategy8	DE−rand1
1000	Best	5.00E−10	1.00E−10	5.33E−04	1.20E−09	2.49E+02	0.00E+00	2.05E+00	0.00E+00	1.00E+00
	Median	5.89E−06	1.00E−10	1.30E−03	5.11E−06	2.82E+02	0.00E+00	2.13E+00	0.00E+00	1.40E+02
	Worst	3.96E−02	1.00E−10	7.96E−02	2.29E−02	3.02E+02	7.44E−02	2.17E+00	0.00E+00	1.71E+02
	Mean	1.62E−03	4.00E−12	6.82E−03	1.21E−03	2.79E+02	2.97E−03	2.12E+00	0.00E+00	1.36E+02
	Std	5.77E−05	3.69E−22	2.45E−04	2.03E−05	2.73E+02	2.12E−04	1.05E−03	0.00E+00	4.81E+02
10000	Best	0.00E+00	1.00E−10	4.66E−05	8.93E−03	7.28E+01	0.00E+00	1.94E+00	0.00E+00	1.70E+01
	Median	0.00E+00	1.00E−10	5.18E−05	4.84E−02	1.61E+02	0.00E+00	2.05E+00	0.00E+00	3.21E+01
	Worst	1.00E−10	1.00E−10	5.44E−05	1.25E+00	2.12E+02	1.00E−10	2.09E+00	0.00E+00	5.68E+01
	Mean	2.80E−11	4.00E−12	5.15E−05	1.71E−01	1.60E+02	8.00E−12	2.04E+00	0.00E+00	3.28E+01
	Std	1.99E−21	3.69E−22	5.51E−12	7.36E−02	1.23E+03	7.07E−22	2.86E−04	0.00E+00	3.39E+02
100000	Best	0.00E+00	1.00E−10	3.94E−05	1.00E−10	4.13E+00	0.00E+00	1.91E+00	0.00E+00	1.59E+01
	Median	0.00E+00	1.00E−10	4.90E−05	1.00E−10	1.11E+01	0.00E+00	1.97E+00	0.00E+00	2.10E+01
	Worst	1.00E−10	1.00E−10	5.21E−05	1.00E−10	1.39E+01	1.00E−10	2.01E+00	0.00E+00	2.86E+01
	Mean	2.80E−11	4.00E−12	4.82E−05	1.00E−10	1.04E+01	4.00E−12	1.97E+00	0.00E+00	2.12E+01
	Std	1.99E−21	3.69E−22	7.00E−10	3.14E−21	3.45E+01	3.69E−22	2.06E−04	0.00E+00	1.33E+02

Table 7 The results obtained by the SaDE and the mixed operator (DE) of automatically generated by the GEP algorithm (AdhDE)

Function	FES		SaDE	AdhDE
F2	1000	Best	5.30E−09	5.00E−10
		Median	9.20E−09	1.52E−08
		Worst	3.38E−08	2.51E−03
		Mean	1.04E−08	1.01E−04
		Std	2.88E−17	2.41E−07
	10000	Best	2.20E−09	0.00E+00
		Median	2.40E−09	5.00E−10
		Worst	2.50E−09	1.51E−07
		Mean	2.38E−09	1.57E−08
		Std	7.74E−21	1.66E−15
	100000	Best	2.30E−09	0.00E+00
		Median	2.40E−09	0.00E+00
		Worst	2.50E−09	1.00E−10
		Mean	2.36E−09	4.40E−11
		Std	3.90E−21	2.46E−21
F5	1000	Best	0.00E+00	0.00E+00
		Median	0.00E+00	0.00E+00
		Worst	1.00E+00	0.00E+00
		Mean	1.60E−01	0.00E+00
		Std	1.34E−01	0.00E+00
	10000	Best	0.00E+00	0.00E+00
		Median	0.00E+00	0.00E+00
		Worst	1.00E+00	0.00E+00
		Mean	3.20E−01	0.00E+00
		Std	2.18 E−01	0.00E+00
	100000	Best	0.00E+00	0.00E+00
		Median	0.00E+00	0.00E+00
		Worst	1.00E+00	0.00E+00
		Mean	2.00E−01	0.00E+00
		Std	1.60E−01	0.00E+00
F8	1000	Best	1.46E+01	0.00E+00
		Median	2.44E+01	1.00E−10
		Worst	4.33E+01	1.50E−09
		Mean	2.62E+01	1.28E−10
		Std	5.92E+01	8.04E−20
	10000	Best	1.69E+01	0.00E+00
		Median	1.89E+01	1.00E−10
		Worst	2.66E+01	1.00E−10
		Mean	2.01E+01	5.60E−11
		Std	7.63E+00	2.46E−21
	100000	Best	1.59E+01	0.00E+00
		Median	2.09E+01	1.00E−10
		Worst	2.39E+01	1.00E−10
		Mean	2.03E+01	5.60E−11
		Std	4.66E+00	2.46E−21

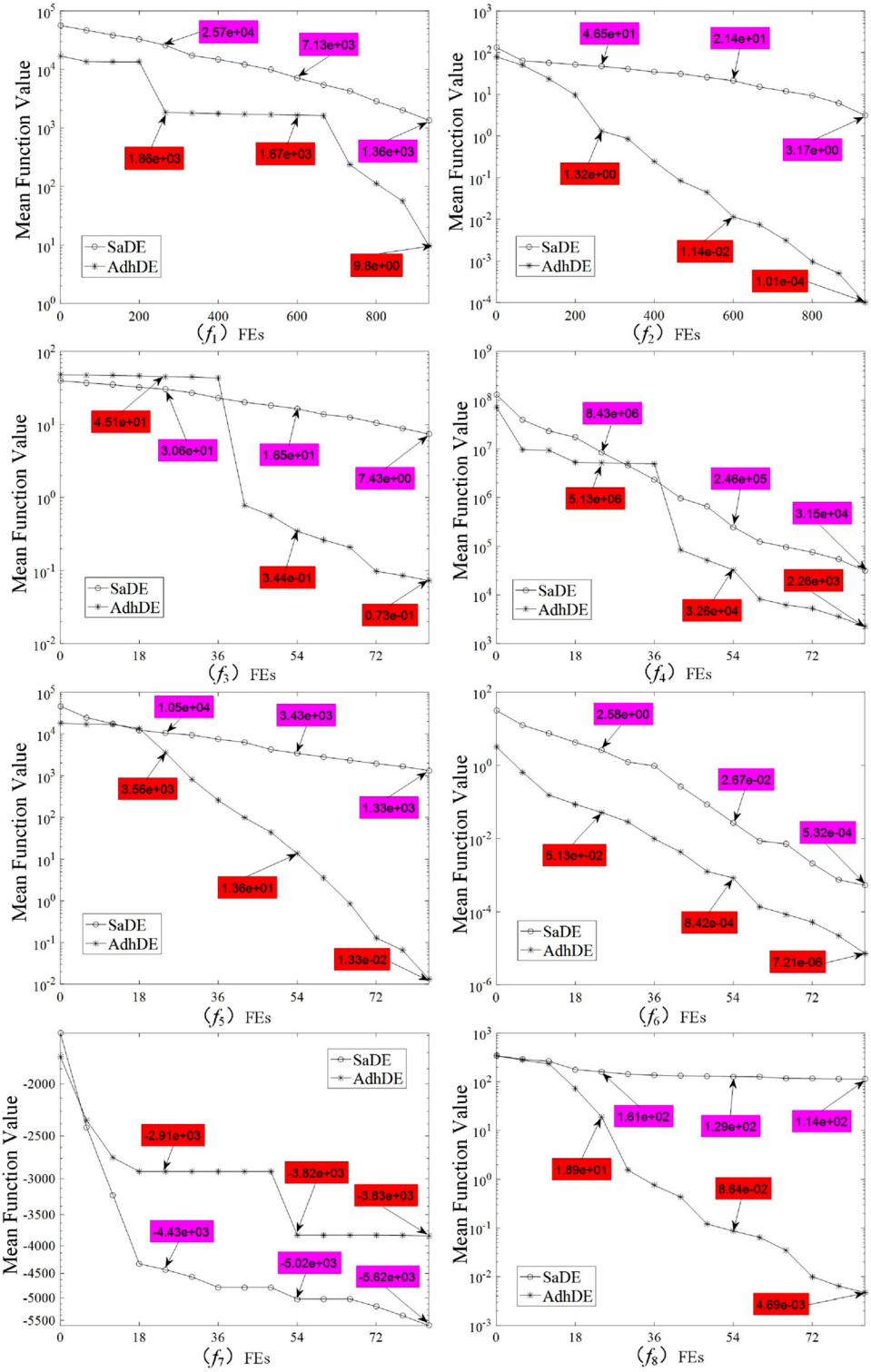

Fig. 5 The convergence speed between the SaDE and AdhDE

most the optimism function than SaDE using genetic operators and has a better effect of the evolutionary algorithm. So this article is divided into three modules: evolutionary algorithm automatically design generating difference strategy, selected candidate difference strategy based on DE algorithm and the hybrid evolutionary algorithm based on DE and compared with the performance of SaDE.

In evolutionary algorithm of automatic design module, the difference strategy based on DE algorithm is generated

automatically by GEP model. Each different difference strategy is automatically generated based on the corresponding problem. Thus, researchers no longer need to manually analyze the data to conclude the corresponding difference strategy. Then different difference strategy was applied to DE algorithm, based on the approximation of solving problems to generate the adaptive value, then through natural selection principle, keeping good individual and constantly optimize the approximate solution, then analyzing the approximation of the solving problems.

Hybrid evolutionary algorithm based on DE, mixed the automatically designed genetic operators, and the minimum value for each function is calculated and validated base on the difference strategy generated. After the verification, we can know the convergence of difference strategy generated by evolution of GEP modeling is better, evolution of GEP modeling can generate effective reasonable difference strategy. DE algorithm based on GEP modeling is also effective in solving most function, and the differential strategy is more adapted in solving corresponding problems to a certain degree.

Acknowledgements The authors would like to thank anonymous reviewers for their very detailed and helpful review. This work was supported by National Natural Science Foundation of China (61902232, 61902231), Natural Science Foundation of Guangdong Province (2019A1515010943), Key Project of Basic and Applied Basic Research of Colleges and Universities in Guangdong Province (Natural Science) (2018KZDXM035), The Basic and Applied Basic Research of Colleges and Universities in Guangdong Province (Special Projects in Artificial Intelligence) (2019KZDZX1030) and 2020 Li Ka Shing Foundation Cross-Disciplinary Research Grant (2020LKSFG04D).

References

Antoniouk AV, Khrennikov AY, Kochubei AN (2019) Multidimensional nonlinear pseudo-differential evolution equation with p-adic spatial variables. J Pseudo Differ Oper Appl 2019(50)

Arram A, Ayob M (2019) A novel multi-parent order crossover in genetic algorithm for combinatorial optimization problems. Comput Ind Eng 133

Brest J, Maučec MS (2011) Self-adaptive differential evolution algorithm using population size reduction and three strategies. Soft Comput 15(11):2157–2174

Chen F, Shi J, Ma Y, Lei Y, Gong M (2017) Differential evolution algorithm with learning selection strategy for SAR image change detection. In: 2017 IEEE congress on evolutionary computation (CEC). IEEE, pp 450–457

Contreras-Bolton C, Parada V (2015) Automatic combination of operators in a genetic algorithm to solve the traveling salesman problem. PloS ONE 10(9):e0137724

Diosan L, Oltean M (2009) Evolutionary design of evolutionary algorithms. Genetic Program Evolv Mach 10(3):263–306

Ferreira C (2001) Gene expression programming: a new adaptive algorithm for solving problems. Complex Syst 13(2):87–129

Goldberg DE (1989) Genetic algorithms in search, Optimization and machine learning, October

Hong L, Drake JH, Woodward JR, Özcan E (2018) A hyper-heuristic approach to automated generation of mutation operators for evolutionary programming. Appl Soft Comput 62:162–175

Ibrahim Abdelmonem M, Tawhid Mohamed A (2019) A hybridization of cuckoo search and particle swarm optimization for solving nonlinear systems. Evolut Intell 2019(6)

Jiang Dazhi Wu, Kaichao Chen Dicheng, Geng Tu, Teng Zhou, Akhil Garg, Liang Gao (2020) A probability and integrated learning based classification algorithm for high-level human emotion recognition problems. Measurements 150:107049

Jiang D, Fan Z (2014) The algorithm for algorithms: an evolutionary algorithm based on automatically designing of genetic operators. Math Probl Eng 2:66–70

Jiang Dazhi, Zhijian Wu, Kang Lishan (2006) New method used in gene expression programming: GRCM. J Syst Simul 18:1466–1468

Jiang D, Peng C, Fan Z (2014) Evolutionary algorithm based on automatically designing of genetic operators. In: 2013 9th international conference on computational intelligence and security. IEEE, pp 66–70

Jiang DT, Geng JD, Kaichao W, Cheng L, Lin Z, Teng Z (2020) A hybrid intelligent model for acute hypotensive episode prediction with large-scale data. Inf Sci

Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of the IEEE international conference on neural networks. IEEE Press, pp 1942–1948

Koza JR (1992) Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge

Liang J, Wang P, Guo L et al (2019) Multi-objective flow shop scheduling with limited buffers using hybrid self-adaptive differential evolution. Memetic Comput (6)

Lin Q, Tang C, Ma Y, Du Z, Li J, Chen J, Ming Z (2017) A novel adaptive control strategy for decomposition-based multiobjective algorithm. Comput Oper Res 78:94–107

Mahanipour A, Nezamabadi-Pour H (2019) GSP: an automatic programming technique with gravitational search algorithm. Appl Intell 49(4):1502–1516

Mallipeddi R, Mallipeddi S, Suganthan PN (2010) Ensemble strategies with adaptive evolutionary programming. Inf Sci 180(9):1571–1581

Nyathi T, Pillay N (2018) Comparison of a genetic algorithm to grammatical evolution for automated design of genetic programming classification algorithms. Expert Syst Appl 104:213–234

Oltean M, Grosan C (2004) Evolving digital circuits using multi expression programming. In: Proceedings. 2004 NASA/DoD conference on evolvable hardware, 2004. IEEE

Preen RJ, Smith J (2019) Evolutionary n-level hypergraph partitioning with adaptive coarsening. IEEE Trans Evolut Comput

Qin AK, Huang VL, Suganthan PN (2009) Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans Evolut Comput 13(2):398–417

Rodriguez-Coayahuitl L, Morales-Reyes A, Escalante HJ (2019) Evolving autoencoding structures through genetic programming. Genetic Program Evolv Mach 2019(8)

Seront G, Bersini H (1996) Simplex GA and hybrid methods. In: Proceedings of IEEE international conference on evolutionary computation, 1996. IEEE

Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11(4):341–359

Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evolut Comput 1(1):67–82

Woodward JR, Swan J (2012). The automatic generation of mutation operators for genetic algorithms. In: Proceedings of the 14th

annual conference companion on genetic and evolutionary computation. ACM, pp 67–74

Xianfang S, Yong Z, Yinan G, Xiaoyan S (2020) Variable-size cooperative coevolutionary particle swarm optimization for feature selection on high-dimensional data. IEEE Trans Evolut Comput. https://doi.org/10.1109/TEVC.2020.2968743

Xiao Guorong, Garg Akhil, Chen Dicheng, Jiang Dazhi (2019) AHE detection with a hybrid intelligence model in smart healthcare. IEEE Access 7(1):37360–37370

Zhang J, Sanderson AC (2009) JADE: adaptive differential evolution with optional external archive. IEEE Trans Evolut Comput 13(5):945–958

Zhong J, Ong YS, Cai W (2015) Self-learning gene expression programming. IEEE Trans Evolut Comput 20(1):65–80