Application of Improved Differential Evolution Algorithm in Solving Equations

Guiying Ning¹ · Yongquan Zhou²

Received: 21 March 2021 / Accepted: 11 November 2021
© The Author(s) 2021

Abstract

The problem of finding roots of equations has always been an important research problem in the fields of scientific and engineering calculations. For the standard differential evolution algorithm cannot balance the convergence speed and the accuracy of the solution, an improved differential evolution algorithm is proposed. First, the one-half rule is introduced in the mutation process, that is, half of the individuals perform differential evolutionary mutation, and the other half perform evolutionary strategy reorganization, which increases the diversity of the population and avoids premature convergence of the algorithm; Second, set up an adaptive mutation operator and a crossover operator to prevent the algorithm from falling into the local optimum and improve the accuracy of the solution. Finally, classical high-order algebraic equations and nonlinear equations are selected for testing, and compared with other algorithms. The results show that the improved algorithm has higher solution accuracy and robustness, and has a faster convergence speed. It has outstanding effects in finding roots of equations, and provides an effective method for engineering and scientific calculations.

Keywords Differential evolution · Convergence · Algebra equation · Fitness · Crossover operator · Mutation operator

1 Introduction

To solve a large number of mathematical problems encountered in scientific and engineering calculations [1, 2], it is often necessary to solve the roots of higher-order algebraic equations or nonlinear equations, or even the roots of transcendental equations or equations [3, 4]. In the long-term research, people have summarized some methods, such as dichotomy, Newton's method, secant method, etc. [5]. However, these traditional methods have high requirements for functions: continuous, differentiable, etc., and they are also very sensitive to the selection of initial values, which brings difficulties to practical solutions. To these problems, in recent years, some scholars have proposed some new methods. For example, Long [6] proposed the derivative-free iterative for solving nonlinear equation, this method

only needs to calculate the function value, and has a second-order convergence rate. Chen et al. [7] proposed clipping method for computing a root of a polynomial, which leads to a higher approximation order and increases the complexity of the calculation. Swarm intelligence algorithm is a strategy and method for solving problems that is developed by simulating the biological evolution process in nature. It is not restricted by search space constraints (such as differentiate, continuous, etc.) and other auxiliary information, and can effectively deal with complex problems that are difficult to solve with traditional mathematical methods. For example, Wang et al. [8] proposed an improved artificial bee colony algorithm for solving nonlinear equations, Zhao et al. [9] used the improvement of artificial glowworm swarm optimization algorithm to solve nonlinear equations. Fan et al. [10] proposed an improved cuckoo search algorithm for solving nonlinear equations. Yong [11] proposed an improved harmony search for nonlinear equations, etc. These algorithms have improved the convergence speed and accuracy to a certain extent, but there are still some problems, such as insufficient diversity, easy to fall into the local extremum, and unable to efficiently find all solutions.

Evolutionary algorithm is a hot issue currently studied, to overcome the shortcomings of evolutionary algorithms

✉ Guiying Ning
guiyingning@126.com

¹ Liuzhou Institute of Technology, Liuzhou 545616, Guangxi, China

² College of Information Science and Engineering, Guangxi University for Nationalities, Nanning 530006, Guangxi, China

in solving the roots of equations: insufficient population diversity leads to premature convergence, falling into local optimum, and low solution accuracy. This paper proposes an improved differential evolution (IDE) algorithm to solve the roots of the equation, the purpose is to provide effective methods for mathematical calculations in science and engineering.

The paper is organized as follows: The basic principles of the DE algorithm and its convergence are described in Sect. 2. The improvement of DE algorithm and the theory of equation roots are discussed in Sect. 3. The numerical experiments are presented in Sect. 4. Conclusions and further research are discussed in Sect. 5.

2 Differential Evolution Algorithm

The DE algorithm was jointly proposed by R. Storn and K. Price [12] in 1995. It is an optimization algorithm based on the swarm intelligence theory. Compared with other evolutionary algorithms such as GA, DE algorithm has a stronger global search strategy. The algorithm uses real number coding, has few control parameters and do not need to rely on the initial feature information of the problem, it is simple and easy to implement. However, according to the principle that there is no free lunch in the world, no matter how good the algorithm is, it cannot solve all problems. Therefore, the approach of scholars is to improve the algorithm for different problems [13–19].

The standard DE algorithm includes crossover, mutation, and selection operations. It is different from other swarm intelligence algorithms in population reproduction schemes. The idea of DE algorithm is to obtain a new individual by adding the weighted difference vector of any two individuals to another individual with certain rules. The generated new individual vector and the current target vector are crossed with a certain probability to generate a new individual vector. In addition, in the selection operation, the DE algorithm uses a greedy way of one-to-one elimination mechanism to select the best individual [20–22]. If the value of the fitness has better than the old one, the new individual will replace it, otherwise the old individual is kept for next generation.

However, with the increase of evolutionary number, the DE algorithm collectively approaches the current optimal solution, resulting in a decline in population diversity and premature convergence.

To overcome these shortcomings, this paper improves the DE algorithm: (1) In terms of improving population diversity, introduce the one-half rule. Combine differential mutation and evolution strategy reorganization; (2) Change the method of taking fixed values in DE, adopt adaptive mutation and crossover operator, to avoid the rapid crowding of

the population to the current optimal solution and lead to premature convergence.

3 Differential Evolution Algorithm for Finding Roots of Equations

In solving scientific and engineering practice, we often encounter the problem of solving roots of higher-order algebraic equations (or systems of equations). In solving these problems, sometimes we have to find the real roots, and sometimes we have to find all the roots. Although many methods have been found to solve these problems. However, most of these methods can only find the real roots, and have strong restrictions on the equation. Therefore, how to quickly find the root of an equation (or systems of equations) has important practical research significance. In this section, we will discuss to solve all the roots of the equations with DE algorithm.

3.1 Distribution Theory of Roots of Algebraic Equations

Given a polynomial equation of degree n :

$$f(z) = a_0 z^n + a_1 z^{n-1} + \dots + a_{n-1} z + a_n \quad (1)$$

The coefficient $a_i (i = 1, 2, \dots, n)$ is real number or complex number. Suppose $z_k (k = 1, 2, \dots, n)$ is the root of $f(z)$, we can choose R and make it satisfy $|z_k| < R (k = 1, 2, \dots, n)$. There are many methods to choose R , according to the usual method, we choose:

$$R = 1 + \max\{|a_k|/|a_0|\} (k = 1, 2, \dots, n)$$

Then suppose $z = Ry$, we can obtain

$$f(Ry) = F(y) = b_0 y^n + b_1 y^{n-1} + \dots + b_{n-1} y + b_n \quad (2)$$

If $y_k (k = 1, 2, \dots, n)$ is the root of $F(y)$, then $|y_k| < 1 (k = 1, 2, \dots, n)$. After finding the roots y_k of $F(y)$, we can get the roots $z_k = Ry_k (k = 1, 2, \dots, n)$ corresponding to $f(z)$. After finding a root y_1 of $F(y)$, use comprehensive division to reduce $F(y)$.

$$F(y)/(y - y_1) \rightarrow F(y), n - 1 \rightarrow n$$

The process is repeated until all roots of $f(z)$ can be found.

The following theorems and conclusion can be found in [23].

Theorem 1 (Basic Theorem of Algebra). Every polynomial with complex coefficients of degree greater than or equal to 1 has one root in the complex number field.

Theorem 2 Every n -degree polynomial of with complex coefficients has exactly n complex roots (multiple roots are calculated by multiplicity).

3.1.1 Conclusion

If the real coefficient polynomial $f(x) = 0$ has complex root $a + bi$, where a and b are both real numbers and $b \neq 0$, then its conjugate $a - bi$ is also the root of the polynomial.

From the theorems and conclusion above, it can be known that any unary n -degree polynomial has n roots in the complex number field. In this paper, an IDE algorithm is used to find all real and complex roots of the polynomial equation.

3.2 Improvement of Differential Evolution Algorithm

The DE generally has premature appearance and slow convergence speed. The main reason is that this algorithm cannot balance the convergence speed and diversity, which is not conducive to the generation of new individuals and dominant individuals. For different stages of evolution, the corresponding optimal parameter values and mutation strategies of DE are also different. To solve these problems. To better develop the performance of the algorithm, an IDE algorithm is designed.

3.2.1 Self-Self-Adaptive Mutation

Mutation is a key operation of DE algorithm. Therefore, the selection of the mutation factor will directly affect the performance of the algorithm. The mutation factor of the basic DE algorithm is to choose a fixed real number between $[0, 2]$. Numerical experiments have proved that during the evolution of the algorithm, the initial mutation factor should be larger to ensure the diversity of the population and avoid precocity. In the later stage, it should be smaller to retain good information, avoid the optimal solution from being destroyed, and ensure its convergence. This paper proposes a self-adaptive mutation operator, which promotes the balance between global search capabilities and local search capabilities. The self-adaptive mutation operator is designed as follows:

$$\eta = \beta(e^\lambda - 1) \quad (3)$$

where $\beta \in [0.2, 0.6]$, $\lambda = T/(T + t)$, T is the maximum iterative number, t is the current iterative number.

Figure 1 shows the evolution curve of the mutation factor in 200 generations. Where $\beta = 0.6$. It can be seen from the figure that the mutation factor η gradually decreases exponentially with the increase of evolutionary algebra. It helps

Fig. 1 Self-self-adaptive mutation evolution curve

to maintain population diversity in the early stage and accelerate the convergence rate in the later stage.

3.2.2 Recombination of Evolutionary Strategy

The median recombination operator of the evolution strategy can make the new individual compatible with the information of the two parents and increase the convergence speed of the algorithm. Therefore, to better maintain the diversity of the population and improve the convergence speed of the algorithm, this paper introduces the evolutionary strategy's median recombination operation during the evolution of the DE algorithm, and the new individuals will participate in the DE crossover operation.

3.3 Specific Steps to Find the Roots of Equations

Step 1. Initialization. Determine the population size M , the crossover probability p_c . And calculate the modulus of the root of the polynomial (1).

Step 2. Randomly select the initial population within $(-1, 1)$. Since the root's modulus of the polynomial (2) is limited to 1, the initial population is randomly generated in $(-1, 1)$.

Step 3. Calculate the fitness of individuals and sort them, select the half with better fitness to participate in population evolution, and retain the best individual in each generation.

Step 4. Mutation. The DE mutation and the median recombination of the evolution strategy is performed to generate new individuals. The calculation formula is as follows:

$$u_{ij}(t+1) = x_{bestj}(t) + \eta(x_{p1j}(t) - x_{p2j}(t))$$

$$w_{ij}(t+1) = (x_{p3j}(t) + x_{p4j}(t))/2 \quad (4)$$

here the subscripts $p1, p2, p3$ and $p4$ are never equal to each other, and $p1, p2, p3$ and $p4$ are arbitrary random integers in $[1, M/2]$. The newly generated individual is denoted as L .

Step 5. Crossover. Crossover operation is conducted as follows:

$$v_{ij}(t+1) = \begin{cases} l_{ij}(t+1)rand \leq p_c \\ x_{ij}(t+1)rand > p_c \end{cases} \quad (5)$$

where p_c is the crossover factor and $p_c \in (0, 1]$, $rand$ is a random number between $[0, 1]$, $i = 1, 2, \dots, M, j = 1, 2, \dots, n$.

Step 6. Selection: Among the new individuals produced in step 5 and the old ones, the 50% optimal individual is selected to form the next generation according to the size of the fitness. Selection operation is conducted as follows:

$$X'_i = \begin{cases} V_i f(V_i) < f(X_i) \\ X_i \text{ otherwise} \end{cases} \quad (6)$$

Step 7. Judgment on conditions. The new individual from selection is denoted as follows:

$$X'(t+1) = (X'_1(t+1), X'_2(t+1), \dots, X'_M(t+1)) \quad (7)$$

The best individual of $X'(t+1)$ is denoted as $X_{\text{best}}(t+1)$. If the given accuracy or the maximum iterative number is satisfied, recording the best individual and outputting $z_k = Rx_{\text{best}k}$, at the same time decline the degree of the polynomial and go on step 2, otherwise set $t = t + 1$ and then go on step 3. The algorithm continues until all the roots of the equation are found.

4 Numerical Experiments

To test the performance of the IDE algorithm proposed in this paper, 12 classic examples were selected for testing. Among these examples, 5 are for finding all the roots of high-order algebraic equations, and 7 are for finding all the roots of nonlinear systems of equations. These examples are from the corresponding literatures.

4.1 Improved Differential Evolution Algorithm to Find Roots of Equations

To test the capability of IDE, five typical model polynomial equations were tested by MATLAB. In this test, the population size is the same as the comparison algorithm, $M = 40$, numerical experiments verify that the effect is best when $\beta = 0.6$, $p_c = 0.3$. The fitness function is the modulus of each equation. The biggest iterative number is 200. Some examples are studied in the literature [24], here only the literature results are given. The results are shown from Tables 1, 2, 3, 4 and 5.

Example 1 [25] Solving polynomial equation $f_1(x) = 0$.

$$f_1(x) = x^8 - 10x^6 + 33x^4 - 40x^2 + 16 = 0$$

Example 2 [26] Finding all the roots of the polynomial equation $f_2(x) = 0$

$$\begin{aligned} f_2(x) = & x^7 + 94.01x^6 + 4406.44005x^5 - 29408.4403x^4 \\ & + 26955.695275x^3 - 12228.972625x^2 \\ & - 123.6375x - 0.625 = 0. \end{aligned}$$

Example 3 [27] Solving polynomial equation $f_3(x) = 0$.

$$\begin{aligned} f_3(x) = & x^{13} + (1+i)x^{12} + ix^{11} + 3ix^{10} \\ & + (7+3i)x^9 + (7+i)x^8 + (i-3)x^7 + (8i-3)x^6 \\ & + (8i-3)x^5 + (7i-3)x^4 + 7ix^3 - 2ix^2 - (8+2i)x - 8 = 0 \end{aligned}$$

Table 3 shows the results of the 200 generations of the IDE algorithm in this paper and the results of 2000 generations of IPSO algorithm [27]. As can be seen from the table, the IDE algorithm uses less time to find a more accurate solution. At the same time, when the evolutionary generation is all 200, the solution results are better than that of the parallel GA [4].

Example 4 [28] Solving polynomial equation $f_4(x) = 0$.

$$\begin{aligned} f_4(x) = & x^8 - (5.5 - 8.5i)x^7 + (28 - 47.75i)x^6 \\ & - (78.75 - 114.25i)x^5 + (142.5 - 58.75i)x^4 \\ & - (23.75 + 134.75i)x^3 - (151.5 - 126.5i)x^2 \\ & + (108 + 12i)x - (20 + 20i) = 0 \end{aligned}$$

Example 5 [25] Solving polynomial equation $f_5(x) = 0$.

$$f_5(x) = x^{41} + x^3 + 1 = 0$$

Table 1 Comparison of IDE and theoretical solutions

IDE solutions	Theoretical solutions	Error accuracy
1.00000000000000 - 0.0000 0000000000i	1	0
- 1.00000000000000 - 0.000 0000000000i	- 1	0
- 1.00000000000000 + 0.000 0000000000i	- 1	0
- 2.00000000000000 - 0.000 0000000000i	- 2	0
2.00000000000000 - 0.00000000000000i	2	0
2.00000000000000 + 0.00000 00000000i	2	0
- 2.00000000000000 + 0.000 0000000000i	- 2	0
1.00000000000000 + 0.00000 00000000i	1	0

Table 2 Comparison of IDE and theoretical solutions

IDE solutions	Theoretical solutions	Error accuracy
$-0.00500000000000 + 0.005000000000i$	$-0.005 + 0.005i$	0
$-0.00500000000000 - 0.005000000000i$	$-0.005 - 0.005i$	0
$0.50000000000000 - 0.500000000000i$	$0.5 - 0.5i$	0
$5.00000000000000 - 0.000000000000i$	5	0
$-50.00000000000001 - 49.9999999999i$	$-50 - 50i$	0.00214862511571
$0.50000000000000 + 0.500000000000i$	$0.5 + 0.5i$	0
$-50.00000000000000 + 49.9999999999i$	$-50 + 50i$	8.053696261997231e-004

Table 3 Comparison of calculation solutions with document solutions on $f_3(x) = 0$

IDE solutions	IPSO solutions	error accuracy	
		IDE	IPSO
$-0.95105651629515 - 0.30901699437495i$	$-0.951056516296339 - 0.30901699437671i$	10^{-14}	10^{-11}
$0.58778525229247 + 0.80901699437495i$	$0.587785252336429 + 0.80901699429743i$	10^{-13}	10^{-9}
$0.95105651629515 - 0.30901699437495i$	$0.9510565159822 - 0.309016994884803i$	10^{-13}	10^{-8}
-1.00000000000000	-1.00000000203754	0	10^{-8}
$-0.58778525229247 + 0.80901699437495i$	$-0.587785252281256 + 0.80901699437671i$	10^{-13}	10^{-10}
$-1.00000000000000i$	$-0.99999999998841i$	0	10^{-9}
$0.86602540378444 - 0.50000000000000i$	$0.86602540372703 - 0.499999999996476i$	10^{-14}	10^{-9}
$1.00000000000000i$	$1.00000000013666i$	0	10^{-8}
$-0.86602540378444 - 0.50000000000000i$	$-0.866025403777964 - 0.50000000000178i$	10^{-14}	10^{-11}
$1.32287565553230 - 1.50000000000000i$	$1.32287565550784 - 1.500000000001638i$	10^{-11}	10^{-7}
$-1.00000000000000 + 1.00000000000000i$	$-0.999999997802094 + 0.99999999879994i$	0	10^{-7}
$-1.32287565553230 - 1.50000000000000i$	$-1.3228756555221 - 1.500000000009083i$	10^{-11}	10^{-7}
$1.00000000000000 + 1.00000000000000i$	$1.00000000000917 + 0.999999999986703i$	0	10^{-8}

Table 4 Comparison of calculation solutions with theories solutions on $f_4(x) = 0$

IDE solutions	Theoretical solutions	Error accuracy
$0.50000000000000 + 0.00000000000000i$	0.5	0
$1.00000000000000 - 0.00000000000000i$	1	0
$-0.00000000000000 + 0.50000000000000i$	0.5i	0
$1.00000000000000 + 1.00000000000000i$	1 + i	0
$-1.00000000000000 + 0.00000000000000i$	-1	0
$2.00000000000000 - 2.00000000000000i$	2 - 2i	0
$0.00000000000000 - 10.00000000000000i$	-10i	0
$2.00000000000000 + 2.00000000000000i$	2 + 2i	0

From the above calculation results, it can be seen that the IDE algorithm can quickly find the root of the equation in the case of real coefficients and complex coefficients, and requires low initial values. It has strong versatility and can find both single and complex roots. From the data in

Table 5, it can also be seen that for higher-order equations, this algorithm has fast calculation speed and stable results, and is significantly better than the standard DE algorithm, it indicates that the algorithm in this paper is robust.

Reference [27] only gives the real root of the problem, and the error of this real root is 10^{-12} , evolution generation is 2000. Our IDE algorithm not only finds all the roots, but also has an error of 10^{-14} corresponding to the real root, which is smaller than the IPSO algorithm [27] and GA [29].

Numerical experiments show that in this paper, the self-adaptive mutation operator and the median recombination operator of the evolution strategy are used in the DE algorithm, and introduce the dichotomy method, greatly improved the performance of the algorithm, and achieved good results in solving all the roots of any high-order algebraic equation with real complex coefficient. It is an effective method for solving all the roots of algebraic equations with real complex coefficients of arbitrary high order.

4.2 Parameter Analysis

This paper analyzes the influence of parameters β and p_c on the algorithm through experiments. According to the parameter setting suggestions of literature [30], use f_1, f_3, f_4 for testing, each set of parameters runs 10 times independently,

Table 5 Comparison of IDE solutions with DE solutions on $f_5(x) = 0$

IDE solutions	DE solutions	Error accuracy	
		IDE	DE
0.41715157825559 – 0.87106730951625i	0.41289430878479 – 0.87779723867229i	10^{-14}	0.1
0.50756863946791 + 0.81057343885272i	0.51664992171016 + 0.81306873744261i	10^{-14}	10^{-2}
– 0.95248387521408 + 0.00000000000000i	– 0.95249212509105 – 0.00001341664792i	10^{-14}	10^{-4}
0.63233982706437 – 0.75340119992382i	0.62610847292127 – 0.75367527711312i	10^{-14}	0.1
– 0.95633901132776 + 0.27377620908702i	– 0.95461170653199 + 0.28062968090088i	10^{-11}	0.1
– 0.18020604323937 + 0.99796223272495i	– 0.17324663258195 + 0.99755957324645i	10^{-11}	0.1
0.98718385803162 + 0.24035438367905i	0.98800969849199 + 0.23204689223504i	10^{-11}	0.1
– 0.33698432323159 – 0.95922761265285i	– 0.34393361434341 – 0.95540052409640i	10^{-11}	0.1
– 0.62067250547060 – 0.80588930705856i	– 0.62599447394494 – 0.79981258744253i	10^{-11}	0.1
0.63233982706403 + 0.75340119992370i	0.64081854381058 + 0.74839180083150i	10^{-12}	0.1
0.93366404473170 – 0.39254638454898i	0.92841007293754 – 0.39843229967857i	10^{-11}	0.1
0.13916543497587 + 0.99247673370054i	0.14462175954003 + 0.98942788151149i	10^{-11}	0.1
– 0.83686300499059 – 0.56782566908490i	– 0.83961912122278 – 0.56025517564735i	10^{-11}	0.1
– 0.73910056519050 + 0.69590435316475i	– 0.73353224894822 + 0.70066366781918i	10^{-11}	0.1
0.28981213106776 – 0.94642367466877i	0.28187124163404 – 0.94990579810634i	10^{-12}	0.1
– 0.33698432323155 + 0.95922761265247i	– 0.32978930402215 + 0.95985364413894i	10^{-11}	0.1
0.98718385803185 – 0.24035438367857i	0.98353281261447 – 0.24745562139318i	10^{-11}	0.1
– 0.96814034151782 – 0.12086695542125i	– 0.96338977824893 – 0.11534518675929i	10^{-12}	0.1
0.93366404473135 + 0.39254638454936i	0.93580420600327 + 0.38435119374059i	10^{-11}	0.1
– 0.18020604323928 – 0.99796223272536i	– 0.18811642350140 – 0.99556940116022i	10^{-11}	0.1
– 0.83686300499034 + 0.56782566908490i	– 0.83234585130459 + 0.57348348599415i	10^{-11}	0.1
0.28981213106724 + 0.94642367466862i	0.29295153694871 + 0.94265506041527i	10^{-11}	0.1
– 0.01972863220789 – 1.00935215687354i	– 0.02791476625721 – 1.00847632676396i	10^{-11}	0.1
0.75371953044317 – 0.65538027600271i	0.74728849875195 – 0.65849582578193i	10^{-12}	0.1
– 0.91051113493335 – 0.42552815131550i	– 0.91119607752174 – 0.41743691504243i	10^{-11}	0.1
1.01430466733546 + 0.08092302985516i	1.01343202866872 + 0.07296387504196i	10^{-11}	0.1
– 0.91051113493315 + 0.42552815131559i	– 0.90719990542079 + 0.43185370498392i	10^{-11}	0.1
0.50756863946789 – 0.81057343885267i	0.50756843083284 – 0.81055259905392i	10^{-13}	10^{-4}
– 0.62067250547036 + 0.80588930705833i	– 0.61416183136871 + 0.80949731706828i	10^{-11}	0.1
0.75371953044270 + 0.65538027600282i	0.75947339113857 + 0.64847105293509i	10^{-11}	0.1
– 0.73910056519075 – 0.69590435316487i	– 0.74311802282295 – 0.68893759606747i	10^{-11}	0.1
– 0.01972863220813 + 1.00935215687315i	– 0.01338436811800 + 1.00725795579966i	10^{-11}	0.1
– 0.95633901132787 – 0.27377620908687i	– 0.95500555216424 – 0.26587682266817i	10^{-11}	0.1
1.01430466733554 – 0.08092302985462i	1.01206265289966 – 0.08882062642299i	10^{-11}	0.1
0.13916543497626 – 0.99247673370085i	0.13088906342826 – 0.99349333402946i	10^{-12}	0.1
0.85515802764341 + 0.53263353549566i	0.85942379905994 + 0.52506974640012i	10^{-11}	0.1
– 0.96814034151785 + 0.12086695542140i	– 0.96954184794274 + 0.12738701929837i	10^{-12}	0.1
0.41715157825506 + 0.87106730951651i	0.41713508592990 + 0.87107150639462i	10^{-12}	10^{-4}
– 0.48528023929995 – 0.89453836983921i	– 0.49169856631144 – 0.88959451611074i	10^{-11}	0.1
– 0.48528023929983 + 0.89453836983888i	– 0.47833517652109 + 0.89683361943375i	10^{-11}	0.1
0.85515802764382 – 0.53263353549541i	0.84970363014098 – 0.53718152172290i	10^{-12}	0.1

count the number of times that all solutions are found, it is stipulated that the solution found is regarded as the optimal solution when the error is less than 10^{-8} . The statistical results are as follows:

It can be seen from Table 6 that when $p_c = 0.3$, the number of times to find all the optimal solutions is the most, when $\beta = 0.6$, for all values of p_c , the number of times is the most. Therefore, in the first part of the experiment, $p_c = 0.3$, $\beta = 0.6$.

4.3 Improved Differential Evolution Algorithm for Solving Nonlinear Equations

To further verify the effectiveness of the IDE algorithm proposed in this paper, seven nonlinear systems of equations [31, 32] were tested. And compared with the literature solutions, this algorithm is programmed with MATLAB. The parameters are also determined by experimental comparison, the experiment found that when $\beta = 0.6$, the crossover

Table 6 Comparison of experimental results of each set of parameters

β	P_c											
	0.2			0.3			0.4			0.6		
	f_1	f_3	f_4	f_1	f_3	f_4	f_1	f_3	f_4	f_1	f_3	f_4
0.2	4	3	5	5	4	6	5	4	6	6	3	7
0.3	5	4	6	7	7	6	6	5	7	5	5	7
0.4	6	6	6	8	7	8	9	6	8	6	6	6
0.5	8	5	7	10	7	9	10	6	9	9	7	10
0.6	8	5	8	10	8	10	10	7	10	8	7	9
0.7	7	6	6	9	8	10	8	7	9	6	8	6
0.8	6	7	8	9	7	9	8	7	8	7	5	7

The bolding is to highlight this set of values

Table 7 The true optimal solutions and searched solutions

Equation	True solutions	IDE searched solutions
Equation 1	$(-1/\sqrt{2}, 1.5)$ (0, 1) (-1, 2)	$(-0.70710678118655, 1.5)$ (0, 1), (-1, 2)
Equation 2	(1.546342, 1.391174) (1.067412, 0.139460)	(1.54634288331995, 1.39117 631279424) (1.06734608580669, 0.13922 766688686)

Fig. 2 The true solutions and the searched solutions of Eq. 3

operator cannot determine the best value. According to [30], it is set to self-adaptively change between [0, 1]. Therefore, the parameter settings are as follows: $M = 40$, $p_c = e^{(t-T)/T}$, $\beta = 0.6$, the fitness function is the sum of the squares of the equations, maximum number of iterations is 300. Each equation runs 20 times and the error is set to 10^{-10} . Table 7 shows the true optimal solutions and searched solutions of Eqs. 1 and 2. Figures 2, 3, 4 and 5 show the comparison

Fig. 3 The true solutions and the searched solutions of Eq. 4 **Fig. 4** The true solutions and the searched solutions of Eq. 5

Fig. 5 The true solutions and the searched solutions of Eq. 6

between the true solutions and the searched solutions from Eqs. 3, 4, 5 and 6.

The tested equations are as follows [31]:

$$\begin{cases} x_1^2 - x_2 + 1 = 0 \\ x_1 - \cos(0.5\pi x_2) = 0 \end{cases} \cdot x \in [-2, 2]. \quad (1)$$

This equation has three true optimal solutions: $(-1/\sqrt{2}, 1.5)$, $(0, 1)$ and $(-1, 2)$ [31].

$$\begin{cases} x_1^2 - x_2 - 1 = 0 \\ (x_1 - 2)^2 + (x_2 - 0.5)^2 - 1 = 0 \end{cases} \cdot x \in [0, 2]. \quad (2)$$

This equation has two true optimal solutions: $(1.546342, 1.391174)$, $(1.067412, 0.139460)$.

For the Eq. 1, the IDE algorithm is better than the HCS algorithm [31], and for the Eq. 2, the two effects are equivalent [31].

$$\begin{cases} x_1^2 + x_2^2 - 2 = 0 \\ x_1^2 + x_2^2/4 - 1 = 0 \end{cases} \cdot x \in [-2, 2]. \quad (3)$$

$$\begin{cases} x_1^4 + 4x_2^4 - 6 = 0 \\ x_1^2 x_2 - 0.6787 = 0 \end{cases} \cdot x_1 \in [-2, 2], x_2 \in [0, 1.1]. \quad (4)$$

These two problems all have four solutions. Table 8 shows the true solutions and the searched solutions by the IDE.

For Eq. 3, the MMODE algorithm [31] can find all the four solutions, but all the searched solutions cannot completely approach the real optimal solution. To Eq. 4, the MMODE can find all solutions. But for the problem with two variables, our IDE algorithm can solve it very well. The HCS algorithm [31] is the second [32].

$$\begin{cases} \sin(x_1^3) - 3x_1 x_2^2 - 1 = 0 \\ \cos(3x_1^2 x_2) - |x_2^3| + 1 = 0 \end{cases} \cdot x_i \in [-2, 2], i = 1, 2. \quad (5)$$

This equation has ten true optimal solutions, see literature [29].

$$\begin{cases} 5x_1^9 - 6x_1^5 x_2^2 + x_1 x_2^4 + 2x_1 x_3 = 0 \\ -2x_1^6 x_2 + 2x_1^2 x_2^3 + 2x_2 x_3 = 0 \\ x_1^2 + x_2^2 - 0.265625 = 0 \end{cases} \cdot x_i \in [-0.6, 0.6], i = 1, 2, x_3 \in [-5, 5]. \quad (6)$$

This problem has 12 real solutions [32].

From the found solutions analysis, for Eq. 5, both the IDE algorithm and the MMODE [32] algorithm can find ten optimal solutions, HCS [31] algorithm has not tested this equation. For Eq. 6 with three variables, the HCS algorithm only finds 11 solutions, and the MMODE algorithm only finds eight optimal solutions, which is not as good as IDE algorithm can find all the optimal solutions. From the calculation time analysis, a hybrid metaheuristic with Fuzzy Clustering Means [33]

Table 8 The true optimal solutions and searched solutions

Equation	True solutions	IDE searched solutions
Equation 3	$(-0.8165, -1.1547)$	$(-0.81649658092773, -1.15470053837925)$
	$(0.8165, -1.1547)$	$(0.81649658092773, -1.15470053837925)$
	$(-0.8165, 1.1547)$	$(-0.81649658092773, 1.15470053837925)$
	$(0.8165, 1.1547)$	$(0.81649658092773, 1.15470053837925)$
Equation 4	$(-1.56353, 0.27763)$	$(-1.56353259158639, 0.27762845244089)$
	$(-0.78971, 1.08830)$	$(-0.78970635977531, 1.08829486020660)$
	$(1.56353, 0.27763)$	$(1.56353259158639, 0.27762845244089)$
	$(0.78971, 1.08830)$	$(0.78970635977531, 1.08829486020660)$

Table 9 The searched solutions by the IDE

Equation	Searched solutions
Equation 5	(− 1.81088519943999, 0.34909099197637) (− 1.81088519943999, − 0.34909099197637) (− 1.79130208461545, − 0.30192634171314) (− 1.50221598606950, 0.40907656829415) (− 0.94726814698626, 0.78502001556829) (− 1.50221598606950, − 0.40907656829415) (− 1.79130208461545, 0.30192634171314) (− 0.94726814698626, − 0.78502001556829) (− 0.21305661923820, − 1.25684531743076) (− 0.21305661923820, 1.25684531743076)
Equation 6	(− 0.51538820320221, 0, − 0.01244559884071) (0, − 0.51538820320221, 0) (0, 0.51538820320221, 0) (− 0.27986557229432, 0.43278189411919, − 0.01418893730114) (0.51538820320221, 0, − 0.01244559884071) (0.27988942827190, − 0.43276656413561, − 0.01418898247693) (0.46697287633137, − 0.21808527845642, 0.00000037696026) (− 0.46674889888061, − 0.21856872352971, 0.00001216788497) (− 0.46697797671607, 0.21807470163761, 0.00000011805959) (0.27985531562219, 0.43278862992836, − 0.01418918511604) (− 0.27967263817071, − 0.43290592324465, − 0.01419015262777) (0.46671963171926, 0.21862570731174, 0.00001429540294)

Table 10 Evolution parameters of each algorithm

Algorithm	Population	Generation
HCS	50 (equation 3–4, 6, 100)	1000 (equation 3–4, 6, 5000)
MMODE	100	500
IDE	40	300

took 0.169 s per execution to find all solutions. A generalized bisection based on boxes [35] took 233 s to find all the solutions. Our method was successful in finding all roots in 0.165 s. Figure 5 shows the comparison between the searched solutions by the IDE and the true solutions (Table 9).

In terms of computational cost, the IDE algorithm in this paper is obviously superior to other algorithms. Table 10 shows the evolutionary generation and population size of each algorithm.

Figures 2, 3, 4 and 5 show the comparison between the searched solutions of Eqs. 3, 4 and 5 and the real solutions.

It can be seen from Figs. 2, 3, 4 and 5 that the solution found by the IDE algorithm in this paper can find all the solutions and is consistent with the real solution. This shows that the improvement of the algorithm in this paper is effective [32].

$$\begin{cases}
 x_1^2 + x_2^2 - 1.0 = 0 \\
 x_3^2 + x_4^2 - 1.0 = 0 \\
 x_5^2 + x_6^2 - 1.0 = 0 \\
 x_7^2 + x_8^2 - 1.0 = 0 \\
 4.731 \times 10^{-3} x_1 x_3 - 0.3578 x_2 x_3 - 0.1238 x_1 + x_7 = 0 \\
 -1.637 \times 10^{-3} x_2 - 0.9338 x_4 - 0.3571 = 0 \\
 0.2238 x_1 x_3 + 0.7623 x_2 x_3 + 0.2638 x_1 - x_7 \\
 -0.07745 x_2 - 0.6734 x_4 - 0.6022 = 0 \\
 x_6 x_8 + 0.3578 x_1 + 4.731 \times 10^{-3} x_2 = 0 \\
 -0.7623 x_1 + 0.2238 x_2 + 0.3461 = 0
 \end{cases} \quad (7)$$

Where $x_i \in [-1, 1]$, $i = 1, 2, \dots, 8$. The problem has 16 solutions. For problems 7 with eight decision variables, MMODE algorithm does not solve well, and its 16 optimal solutions are not well searched, which show that MMODE still needs improvement in solving problems with many decision variables. The HCS algorithm finds its 11 solutions, which shows that the HCS algorithm is not perfect for solving nonlinear equations with a large number of solutions, and the coverage of the solutions in the iterative process is not complete, and the diversity of the solutions is not sufficient. A hybrid metaheuristic with Fuzzy Clustering Means [33] took an average time of 0.295 s per execution to find all solutions. Kearfott [34] took 1120 s to find all the roots. Our method was successful in finding all roots taking an average time of 0.276 s per execution. The calculation results show that $\times 1, \times 2, \times 3$ have four different values, $\times 4, \times 5, \times 6$ have 16 different values, and $\times 7, \times 8$ have eight different values. Due to the dimensional relationship, the corresponding solutions are marked with different signs in Fig. 6. Figure 6 shows the comparison between the 16 solutions found by our IDE algorithm and the real solutions.

Fig. 6 The true solutions and the searched solutions of Eq. 7

Fig. 7 Shape of beam section [35]

Table 11 The sensitivity of the equation to unknowns

(h, b, t)	Error of Eq. 1	Error of Eq. 2	Error of Eq. 3
(22.8949, 12.5655, 2.7898)	1.7229	175.2574	378.0518
(22.8969, 12.3566, 2.7890)	0.5299	57.4324	121.9666
(22.8969, 12.2566, 2.7886)	-0.0471	-0.0597	0.0168

For Eqs. 6 and 7 with multiple variables, our IDE algorithm in this paper can also find all the optimal solutions to the problem in a short time. Moreover, the calculation speed is faster than the comparison algorithm. This shows that IDE algorithm can guarantee the diversity of the population in the iterative process, and can also perfectly solve the nonlinear systems of equations with a large number of solutions.

5 Application of Improved Differential Evolution Algorithm in Engineering

Figure 7 is a cross-sectional view of a thin-walled rectangular beam that is common in architectural design. Knowing the load of thin-walled rectangular beams, it is required to determine the size of the beam section.

The nonlinear equation system of this problem is:

$$\begin{cases} A = bh - (b - 2t)(h - 2t) \\ I_y = bh^3/12 - (b - 2t)(h - 2t)^3/12 \\ I_n = 2(h - t)^2(b - t)^2t/(h + b - 2t) \end{cases}$$

Among them, b is the width of the rectangular beam section, h is the height, and t is the thickness.

Although this system of nonlinear equations is only three-dimensional, it is a relatively classic problem, and it hardly converges with Newton's iteration method.

When $A = 165$, $I_y = 9369$, $I_z = 6835$, each equation of this system of equations is very sensitive to the value of the unknown, as shown in Table 11.

Due to the sensitivity of the equation to unknowns, when using an optimization method to solve this system of equations, the algorithm is required to have strong global search capabilities.

Run 100 times under the same conditions to calculate the success rate, where the success rate is: $\text{error} = \max |f_i(x)| < 10^{-6}$. Table 12 shows the comparison of the convergence reliability of various algorithms.

The optimal solution found by the IDE algorithm that meets the engineering application is: $h=22.89493803$, $b=12.25651911$, $t=2.78981782$.

Through this example, it can be shown that the IDE algorithm can well overcome the defect that the DE algorithm is easy to fall into the local extreme value. The optimization function that is sensitive to variables shows stronger optimization ability, so that the method can also find satisfactory solutions to the equations in engineering applications.

6 Conclusion and Future Work

Solving the roots of high-order real complex coefficient algebraic equations (systems of equations) is of great significance to the analysis and comprehensive design of engineering projects. This paper used self-adaptive mutation operator and cross operation in DE algorithm and introduced the median recombination operator of the evolution strategy. And dichotomy was introduced during evolution. The proposed method is simple to operate and easy to implement. Using the IDE algorithm to solve all roots of high-order algebraic equation with real complex coefficient, some classic equations and nonlinear systems of equations with multiple solutions are tested.

Table 12 Contrast of convergence reliability

Newton method	PSO [36]	MC [36]	MCP SO [36]	GA [37]	HGA [37]	IDE
42%	71%	39%	98%	0%	100%	98%

The experimental results clearly show that the solutions of all equations and nonlinear systems of equations can be found completely. It shows that the IDE algorithm can effectively enhance the diversity of the population and the robustness of the algorithm. It can deal with equations in engineering applications. In addition, it is an optimized calculation technique, and related mathematical equation models in other fields can also be dealt with.

In the future, we will further enhance the stability of the IDE by employing some advanced strategies and combining other optimization algorithms, and apply our methodology to other nonlinear systems of equations, as for example more real-world examples and the real-world multimodal optimization problems in general.

Acknowledgements This work was supported by the Science and Technology Research Project of Guangxi Universities (KY2015YB521), the Youth Education Teachers' Basic Research Ability Enhancement Project of Guangxi Universities (2019KY1098). The authors are grateful to all the anonymous reviewers for their suggestions and remarks that were decisive in improving this article.

Declarations

Conflict of interest The authors declare that they have no conflicts of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Cao, Z.H., Zhang, Y.D., Li, R.X.: Matrix computing and equations solving. People Education Press, Beijing (1979)
2. Department of Applied Mathematics, National Chiayi University. Numerical analysis study networks [EB/OL]. <http://www.math.ncyu.edu.tw/enum/>. Accessed 31 May 2005
3. Liu, F., Chen, G.L., Wu, H.: Design and implementation of genetic algorithm for finding roots of complex functional equation. *Control Theor. Appl.* **21**(3), 467–469 (2004)
4. Liu, F., Chen, G.L., Wu, H.: Parallel genetic algorithm finding roots of complex functional equation based on PVM. *Mini Micro Syst.* **24**(7), 1358–1361 (2003)
5. Wang, X.Y., Fang, D.H.: An improvement on solving nonlinear equation by genetic algorithm. *J. Jishou Univ. (Nat. Sci. Ed.)* **28**(4), 35–38 (2007)
6. Long, A.F.: The derivative free iterative for solving nonlinear equation. *College Math.* **33**(2), 108–110 (2017)
7. Chen, X.D., Ma, W.: A planar quadratic clipping method for computing a root of a polynomial in an interval. *Comput. Graph.* **46**(C), 89–98 (2015)
8. Wang, J.W., Yang, D., Qiu, J.F., et al.: Improved artificial bee colony algorithm for solving nonlinear equations. *J. Anhui Univ. (Nat. Sci. Ed.)* **38**(3), 16–23 (2014)
9. Zhao, F.W., Zhou, Y.Q.: Using the improvement of artificial glow-worm swarm optimization algorithm to solve nonlinear equations. *Math. Pract. Theor.* **46**(1), 176–186 (2016)
10. Fan, X.M., Xu, T.L.: Improved cuckoo search algorithm for solving nonlinear equations. *J. Inner Mongolia Normal Univ. (Nat. Sci. Ed.)* **46**(1), 13–16 (2017)
11. Yong, L.Q.: An improved harmony search for nonlinear equations. *J. Chongqing Univ. Technol. (Nat. Sci.)* **34**(10), 231–237 (2020)
12. Storn, R., Price, K.: Differential evolution—a simple and efficient self-adaptive scheme for global optimization over continuous spaces. *J. Global Optim.* **11**(4), 341–359 (1997)
13. Dey, A., Agarwal, A., Dixit, P., et al.: Genetic algorithm for total graph coloring. *J. Intell. Fuzzy Syst.* (2019). <https://doi.org/10.3233/JIFS-182816>
14. Dey, A., Broumi S., Son L. H., et al. A new algorithm for finding minimum spanning trees with undirected neutrosophic graphs. *Granular Computing.* (2018). <https://doi.org/10.1007/s41066-018-0084-7>
15. Wu, J.J., Song, Y.J., Zhao, H.M.: Differential evolution algorithm with wavelet basis function and optimal mutation strategy for complex optimization problem. *Appl. Soft Comput.* (2020). <https://doi.org/10.1016/j.asoc.2020.106724>
16. Mohanta, K., Dey, A., Pal, A., et al.: A study of m-polar neutrosophic graph with applications. *J. Intell. Fuzzy Syst.* **4**(38), 4809–4828 (2020)
17. Broumi, S., Dey, A., Talea, M., et al.: Shortest path problem using bellman algorithm under neutrosophic environment. *Complex Intell. Syst.* **5**(4), 409–416 (2019)
18. Dey, A., Son, L.H., Pal, A., et al.: Fuzzy minimum spanning tree with interval type 2 fuzzy arc length: formulation and a new genetic algorithm. *Soft Comput.* **6**, 1–12 (2019)
19. Mohanta, K., Dey, A., Pal, A.: A note on different types of products of neutrosophic graphs. *Complex Intell. Syst.* (2020). <https://doi.org/10.1007/s40747-020-00238-0>
20. Yang, W.Q., Cai, L., Xue, Y.C.: A survey of differential evolution algorithm. *PR&AI* **4**(8), 506–511 (2008)
21. Duan, Y.H., Gao, Y.L.: A particle swarm optimization algorithm based on differential evolution. *Comput. Simul.* **26**(6), 212–245 (2009)
22. Yu, Q., Zhao, H.: Network projection model based on differential evolution algorithm and its application. *Comput. Eng. Appl.* **44**(14), 246–248 (2008)
23. Wang, E.F., Shi, S.M.: Higher algebra, 3rd edn. Higher Education Press (2004)
24. Ning, G.Y., Zhou, Y.Q.: Improved differential evolution algorithm for finding all roots of equations. *Comput. Eng. Design* **29**(12), 3173–3176 (2008)
25. Ma, Z.H., et al.: Handbook of modern applied mathematics, volumes of calculation and numerical analysis. Tsinghua University Press, Beijing (2005)
26. Wang, X.L.: Optimal binomial factor for Nth degree algorithm equation with real coefficients. *J. Harbin Eng. Univ.* **1**(2), 109–112 (2004)
27. Gao, F., Tong, H.Q.: Improved particle swarm optimization algorithm for finding roots of equations. *Wuhan Univ. (Nat. Sci. Ed.)* **3**(6), 296–300 (2006)
28. Chen, J.S.: Stability judgment and polynomial root algorithm[J]. *Chin. J. Appl. Math.* **26**(4), 316 (2003)

29. Chen, Z.Y., Kang, L.S.H., Hu, X.: Application of genetic algorithm for solving equation. *Wuhan Univ. (Nat. Sci. Ed.)* **44**, 577–580 (1998)
30. Bošković, B., Mernik, M.: Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. *IEEE Trans. Evol. Comput.* **12**(6), 645–657 (2006)
31. Li, X.F., Zhong, M.H., Zheng, H.Q.: Hybrid cuckoo search algorithm for solving nonlinear equations. *Math. Practice Theor.* **11**(6), 197–204 (2019)
32. Xu, W.W., Ling, J., Yue, C.T., et al.: Multimedia multi-objective differential evolution algorithm for solving nonlinear equations. *Appl. Res. Comput.* **5**(5), 1306–1309 (2019)
33. Sacco, W.F., Henderson, N.: Finding all solutions of nonlinear systems using a hybrid metaheuristic with fuzzy clustering means. *Appl. Soft Comput.* **11**, 5424–5432 (2011)
34. Kearfott, R.B.: Some tests of generalized bisection. *ACM Trans. Math. Softw.* **13**, 197–220 (1987)
35. Chen, ZY.: Research on particle swarm algorithm and its engineering application. University of Electronic Science and Technology of China, Chengdu, 59–60 (2007)
36. Mo, YB.: The extended forms of particle swarm optimization algorithm and their application. Zhe Jiang University, HangZhou 76–78 (2006)
37. Luo, Y.Z.H., Yuan, D.C., Tang, G.J.: Hybrid genetic algorithm for solving systems of nonlinear equations. *Chin. J. Comput. Mech.* **22**(1), 1–6 (2005)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.