Improved cuckoo optimization algorithm for solving systems of nonlinear equations

Article *in* The Journal of Supercomputing · March 2016

DOI: 10.1007/s11227-016-1660-8

3 authors:

Mahdi Abdollahi
Victoria University of Wellington

Asgarali Bouyer
Azarbaijan Shahid Madani University

Davoud Abdollahi
University of Tabriz

J Supercomput
DOI 10.1007/s11227-016-1660-8

Improved cuckoo optimization algorithm for solving systems of nonlinear equations

Mahdi Abdollahi[1] · **Asgarali Bouyer**[2] · **Davoud Abdollahi**[3]

Abstract Systems of nonlinear equations come into different range of sciences such as chemistry, economics, medicine, robotics, engineering, and mechanics. There are different methods for solving systems of nonlinear equations such as Newton type methods, imperialist competitive algorithm, particle swarm algorithm, conjugate direction method that each has their own advantages and weaknesses such as low convergence speed and poor quality of solutions. This paper improves cuckoo optimization algorithm for solving systems of nonlinear equations by changing the policy of egg laying radius, and some well-known problems are presented to demonstrate the efficiency and better performance of this new robust optimization algorithm. From obtained results, our approach found more accurate solutions with the lowest number of function evaluations.

Keywords Cuckoo optimization algorithm · Evolutionary algorithms · Nonlinear equations · Optimization

✉ Mahdi Abdollahi
abdollahi_mm@yahoo.com; m.abdollahi89@ms.tabrizu.ac.ir

Asgarali Bouyer
a.bouyer@azaruniv.edu

Davoud Abdollahi
abdollahi_d@daneshvaran.ac.ir

1 Department of Computer Engineering, Miandoab Branch, Islamic Azad University, Miandoab, Iran

2 Faculty of Computer Engineering and Information Technology, Azarbaijan Shahid Madani University, Tabriz, Iran

3 University College of Daneshvaran, Tabriz, Iran

Published online: 06 February 2016

1 Introduction

Solving systems of nonlinear equations has always been important in science. Most of the scientific problems are related to the system of nonlinear equations. There are two types of system equations. The first type is linear and the second type is called nonlinear. There are several methods for the first type but there are few methods for the second type that the solution often comes with approximation.

So far, many methods have been presented for solving the systems of nonlinear equations problems such as El-Emary and El-Kareem, who employed Gauss–Legendre integration as a technique to solve the system of nonlinear equations and used genetic algorithm (GA) to find the results without converting the nonlinear equations to linear equations [1]. Wu and Kang who used a parallel elite-subspace evolutionary algorithm (PEA) [2]. Mastorakis, who employed genetic algorithm to solve a non-linear equation as well as systems of non-linear equations [3]. Li and Zeng, who used a neural-network algorithm for solving a set of nonlinear equations.

The computation is carried out by a simple gradient descent rule with variable step-size levels [4]. Huan-Tong et al. who proposed a modified evolution strategy based on probability ranking method to solve complicated nonlinear systems of equations [5]. Abdollahi et al. applied imperialist competitive algorithm (ICA) for solving nonlinear systems equations [6]. Ouyang et al. employed a hybrid particle swarm optimization algorithm. The particle swarm optimization (PSO) method focuses on "exploration" and the Nelder-Mead simplex method focuses on "exploitation" [7]. Wu et al. used a new variation of the Social emotional optimization algorithm called MSEOA, mainly on the thought of Metropolis Rule [8]. Luo at al. applied a combination of chaos search and Newton type methods [9]. Grosan and Abraham employed a new perspective of evolutionary algorithm [10]. Mo et al. proposed a combination of the conjugate direction method [11]. Jaberipour used the proposed particle swarm algorithm (PPSO) [12]. Oliveira and Petraglia, who proposed the fuzzy adaptive simulated annealing (ASA) for solving nonlinear systems of functional equations [13]. Henderson et al. [14] and Pourjafari et al. [15] introduced a methodology via a polarization technique and a novel optimization method based on Invasive Weed Optimization (IWO)respectively for finding all roots of systems of nonlinear equations.

The obtained answers of mathematical methods are sensitive to the initial guess of the solution [9,11]. The population size of the evolutionary algorithms (GA [1], EGA [3], EA [10], PEA [2], MES [5], MSEOA [8], PSO [7], PPSO [12], ASA [13], ICA [6] and IWO [15]) is large and the convergence of the mentioned methods to the global minimum is slow. For this reason, it is necessary to find an efficient algorithm for solving the systems of nonlinear equations. In the current paper, the problem is finding parameter x for solving (1):

$$f(x) = 0 \tag{1}$$

The square of function (1) is as follows:

$$F(x) = f^2(x) \geq 0 \tag{2}$$

Finding the absolute minimum for $F(x)$ in (2) can be a solution to (1). Another method that can be used, as follows:

$$F(x) = |f(x)| \geq 0 \tag{3}$$

Another method to solve (1) is the method presented by Anjel Kuri–Morals [11]. In this method min $F(x)$ considered under the constraint $f(x) \geq 0$ (or $f(x) \leq 0$). Now, the mentioned idea can be used for a system of n equations in n unknown variables. Let the form of systems of nonlinear equations be:

$$\begin{cases} f_1(x_1,\ x_2,\ \ldots,\ x_n) = 0 \\ f_2(x_1,\ x_2,\ \ldots,\ x_n) = 0 \\ \vdots \\ f_n(x_1,\ x_2,\ \ldots,\ x_n) = 0 \end{cases} \tag{4}$$

The square function and the absolute value function of (4) are defined as $F(x) = f_1^2 + f_2^2 + \cdots + f_n^2$ and $F(x) = |f_1| + |f_2| + \cdots + |f_n|$ respectively. If the absolute minimum for the function $F(x)$ is zero at the point $x_1^*,\ x_2^*,\ \ldots,\ x_n^*$ then $x_1^*,\ x_2^*,\ \ldots,\ x_n^*$ is a solution of (4). Anjel Kuri–Morals developed another method for solving (4) by function $min(f_1 + f_2 + \cdots + f_n)$ under the constraints $f_1(x) \geq 0,\ f_2(x) \geq 0,\ \ldots,\ f_n(x) \geq 0$ [16]. Table 1 shows all of the mentioned methods.

According to the presented results for evaluating the three mentioned methods in [17], the first method shows better performance. So in order to transform (4) to an optimization problem, we will use the auxiliary function:

$$\min F(x) = \sum_{i=1}^{n} f_i^2(x), \quad x = (x_1, x_2, \ldots, x_n) \tag{5}$$

where $F(x)$ is the cost function that will be minimized.

In this paper, we introduce an improved cuckoo optimization algorithm (iCOA) for solving systems of nonlinear equations problems that solves the mentioned weaknesses of the mathematical and evolutionary methods. Some well-known problems are selected to evaluate the method. Then, we compare the results with COA and some other approaches to solve the same problems.

The rest of the paper is organized as follows: Sect. 2 presents the cuckoo optimization algorithm (COA) and describes the idea that improves the COA. In Sect. 3,

Table 1 The common methods for handling systems of nonlinear equations to an optimization problem

Method	Cost function
First	$F(x) = f_1^2 + f_2^2 + \cdots + f_n^2$
Second	$F(x) = \|f_1\| + \|f_2\| + \cdots + \|f_n\|$
Third	$F(x) = f_1 + f_2 + \cdots + f_n$

we compare the obtained results with previously proposed methods. At the end, the conclusions and future works are presented in Sect. 4.

2 Cuckoo optimization algorithm

Cuckoo optimization algorithm was introduced by Rajabioun [18] that was inspired by the lifestyle of a bird family called cuckoos. In order to solve the problems, the values of variables must be an array. This array is called "Country" and "Particle Position" in ICA and PSO, respectively, and shows a solution of optimization problem. In COA the mentioned array is called a "habitat". So a habitat is a one-dimensional array ($1 \times N_{\text{var}}$) that indicates the living location of a cuckoo. The cost of a habitat is gained by evaluation of fitness function. We should mention that COA maximizes the fitness function. So for the problems that should be minimized, one can easily multiple the fitness function to a minus.

The important parts of COA are egg laying and reproduction of offsprings. Like all evolutionary algorithms, the COA starts with an initial population of cuckoos (N_{pop}) that is generated randomly. Therefore, the population is a matrix of size $N_{\text{pop}} \times N_{\text{var}}$. Then a number of eggs that are produced randomly are assigned for each of cuckoo habitats. In this algorithm, each cuckoo has a range for egg laying that is showed with a lower limit (var_{low}) and upper limit (var_{hi}). Thus, the number of eggs that are produced randomly depends on the lower and upper bounds, that is assigned for each of cuckoo habitats. This number is generated for any cuckoo at all iterations. In nature, real cuckoos lay eggs within a maximum distance from their habitats. This maximum range is called "Egg Laying Radius (ELR)". Each cuckoo has an ELR which is defined as:

$$\text{ELR} = \alpha \times \frac{\text{Number of current cuckoo's eggs}}{\text{Total number of eggs}} \times (var_{\text{hi}} - var_{\text{low}}) \tag{6}$$

where α is an integer number, supposed to handle the maximum value of ELR.

After assigning ELR for each cuckoo, all of them start laying eggs randomly in some other host birds' nests that are within cuckoo's ELR. When the egg laying process is finished, $p\%$ of all eggs (usually 10 %) that have less cost values, will be killed by host birds. After the young cuckoos grow and become adult, for having more opportunity to live, they have to immigrate to new habitats that contain more cost values than others. This raises the chance of survival and growth for them. In nature, cuckoos live in separate groups. To determine which cuckoo belongs to which group, the k-means clustering method is used. K is defined as depending on the size of the search space. That is, the value of k increases for large spaces and decreases for small spaces.

When the cuckoos start to move towards the goal point, they only fly a part of the distance with a small deviation. Each cuckoo only flies $\lambda\%$ of the all way towards the goal habitat with a deviation of φ radians. This helps cuckoos to search the space more efficiently. For each cuckoo, λ and φ are defined as follows:

$$\begin{aligned} \lambda &\sim U(0,\ 1) \\ \varphi &\sim U\left(-\frac{\pi}{6},\ \frac{\pi}{6}\right) \end{aligned} \tag{7}$$

In real life, there are food and space constraints to survive. To simulate this fact and to have balance in cuckoos population, the variable $N_{\max}$ is used to control the number of live cuckoos in the nature.

After some iterations, all the cuckoos will move to a nest which has more profit than the other places. In this nest, eggs have the maximum similarity with more food resources and less egg losses in comparison with other nests. When more than 95 % of all cuckoos converge to the same habitat, COA will be terminated.

2.1 Improved cuckoo optimization algorithm

According to the main COA, the migration operator and egg laying operator play the role of a global search and a local search respectively and α of ELR in (6) is fixed during each process, so, in some problems, especially in the systems of nonlinear equations, COA is not able to find the accurate solution and only finds the solutions that are close to the global optimum. If we set a large size to the radius, the COA in early iterations converges very rapidly to the vicinity of the global optimum, but in the next iterations it can't find the accurate answer. If we set a small size to the radius, the COA falls in the local optimum. To solve this problem, the constant α is decreased by the following formula:

$$\alpha_{\text{Iter}} = \alpha_{\text{Iter}-1} \times \beta \tag{8}$$

where β is a number on (0, 1), supposed to control the value of ELR, so, the new ELR defines as follows:

$$\text{ELR} = \alpha_{\text{Iter}} \times \frac{\text{Number of current cuckoo's eggs}}{\text{Total number of eggs}} \times 1 \tag{9}$$

The constant $var_{\text{hi}} - var_{\text{low}}$ supposed 1. This makes it easier to control the radius. In the new COA method that is called "the improved Cuckoo Optimization Algorithm", for having an accurate local search, the egg laying rate must be decreasing gradually from the maximum value of α to 0 with a linear decreasing rate by β in (8). In this case, the iCOA with larger radius will find the answer that is close to the global optimum and by getting smaller radius, it will be able to find the exact solution of the problem. Figure 1 shows the effect of beta coefficient on the ELR during the execution of iCOA.

The termination state of the iCOA happens in three cases. First, it can happen after a certain number of iterations. Second, it can occur after a specific number of fitness

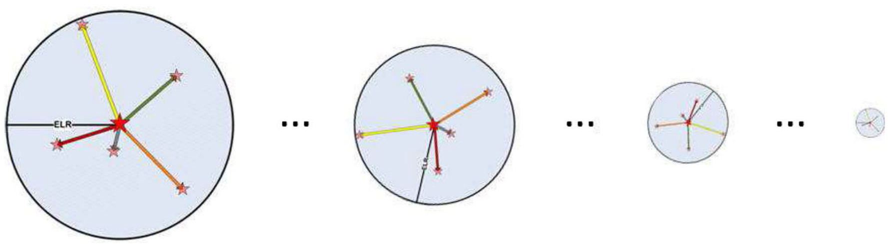

Fig. 1 The effect of beta coefficient on the ELR during the execution of iCOA

1. *Initialize cuckoo habitats with some random points on the profit function*
2. *Dedicate some eggs to each cuckoo*
3. *Compute the value of the ELR's coefficient (α) by the β coefficient*
4. *Define ELR for each cuckoo*
5. *Let cuckoos to lay eggs inside their corresponding ELR*
6. *Kill those eggs that are recognized by host birds*
7. *Let eggs hatch and chicks grow*
8. *Evaluate the habitat of each newly grown cuckoo*
9. *Limit cuckoos' maximum number in environment and kill those who live in worst habitats*
10. *Cluster cuckoos and find best group and select goal habitat*
11. *Let new cuckoo population immigrate toward goal habitat*
12. *if stop condition is satisfied stop, if not go to 2*

Fig. 2 Pseudo-code for improved cuckoo optimization algorithm

Table 2 Used parameters in iCOA for tests and cases

Parameters	Test 1 ($D = 10$)	Test 2 ($D = 100$)	Test 2	Case 1	Case 2	Case 3	Case 4	Case 5
Initial pop.	5	5	5	5	5	5	5	5
Range of eggs	[2, 4]	[2, 4]	[2, 6]	[2, 6]	[2, 5]	[2, 5]	[2, 5]	[2, 5]
Motion coefficient	9	9	0.5	2	9	9	30	5
Maximum of cuckoos	10	80	30	20	15	30	30	40
Number of clusters	1	1	1	1	1	1	1	1
Pop. variance	1e−10	1e−13	1e−13	1e−100	1e−100	1e−100	1e−100	1e−100
α	0.1	0.09	130	10	0.0001	0.01	0.01	1
β	0.9	1	0.99	0.85	0.8	0.89	0.84	0.9

Table 3 Results of Mo et al. (from [11])

Ordinal number	Optimal solution $(x_1, x_2, x_3, x_4, x_5, x_6)$	Optimal value	Iteration iterations	Mean iteration of 10 runs
1	(0.2031, 0.1479, 0.4767, 0.2753, 0.3116, 0.6573)	−3.3220	79	137.5
2	(0.2030, 0.1469, 0.4758, 0.2756, 0.3120, 0.6572)	−3.3220	74	
3	(0.2019, 0.1455, 0.4766, 0.2754, 0.3112, 0.6573)	−3.3220	227	
4	(0.2022, 0.1475, 0.4772, 0.2752, 0.3115, 0.6568)	−3.3220	86	
5	(0.2018, 0.1468, 0.4774, 0.2755, 0.3122, 0.6582)	−3.3220	221	
6	(0.2031, 0.1479, 0.4767,0.2755, 0.3116, 0.6573)	−3.3220	79	
7	(0.2030, 0.1469, 0.4758, 0.2756, 0.3120, 0.6572)	−3.3220	74	
8	(0.2019, 0.5455, 0.4766, 0.2754, 0.3112, 0.6573)	−3.3220	227	
9	(0.2022, 0.1475, 0.4772, 0.2752, 0.3115, 0.6568)	−3.3220	86	
10	(0.2018, 0.1468, 0.4774, 0.2755, 0.3122, 0.6582)	−3.3220	220	

function evaluations (NFEs). Finally, it can terminate when the value of ELR be 0. In the last case, the algorithm could not continue because there is no any radius for egg laying by the cuckoos. In the current paper, the number of function evaluations is the

Table 4 Results of ICA (from [19])

Ordinal number	Optimal solution $(x_1, x_2, x_3, x_4, x_5, x_6)$	Optimal value	Iteration iterations	Mean iteration of 10 runs
1	(0.2023, 0.1458, 0.4753, 0.2754, 0.3118, 0.6574)	−3.3220	47	83.3
2	(0.2021, 0.1475, 0.4756, 0.2760, 0.3115, 0.6574)	−3.3220	88	
3	(0.2012, 0.1467, 0.4785, 0.2755, 0.3119, 0.6570)	−3.3220	89	
4	(0.2017, 0.1467, 0.4784, 0.2752, 0.3117, 0.6573)	−3.3220	97	
5	(0.2014, 0.1455, 0.4771, 0.2750, 0.3112, 0.6573)	−3.3220	72	
6	(0.2017, 0.1472, 0.4763, 0.2746, 0.3116, 0.6572)	−3.3220	96	
7	(0.2030, 0.1469, 0.4774, 0.2758, 0.3115, 0.6573)	−3.3220	85	
8	(0.2004, 0.1470, 0.4759, 0.2748, 0.3119, 0.6575)	−3.3220	73	
9	(0.2026, 0.1471, 0.4754, 0.2750, 0.3117, 0.6572)	−3.3220	87	
10	(0.2016, 0.1468, 0.4785, 0.2756, 0.3112, 0.6574)	−3.3220	99	

Table 5 Results of COA

Ordinal number	Optimal solution $(x_1, x_2, x_3, x_4, x_5, x_6)$	Optimal value	Iteration iterations	Mean iteration of 10 runs
1	(0.2021, 0.1463, 0.4764, 0.2761, 0.3115, 0.6565)	−3.3220	101	155.8
2	(0.2011, 0.1466, 0.4769, 0.2757, 0.3122, 0.6572)	−3.3220	96	
3	(0.2016, 0.1462, 0.4763, 0.2760, 0.3120, 0.6568)	−3.3220	89	
4	(0.4052, 0.8821, 0.9363, 0.5741, 0.1369, 0.0379)	−3.3220	125	
5	(0.2015, 0.1479, 0.4767, 0.2745, 0.3115, 0.6571)	−3.3220	103	
6	(0.4046, 0.8826, 0.8378, 0.5740, 0.1260, 0.0395)	−3.3220	500	
7	(0.2022, 0.1462, 0.4754, 0.2748, 0.3114, 0.6576)	−3.3220	96	
8	(0.2021, 0.1463, 0.4780, 0.2759, 0.3121, 0.6572)	−3.3220	69	
9	(0.4041, 0.8823, 0.8741, 0.5734, 0.1452, 0.0395)	−3.3220	169	
10	(0.4042, 0.8819, 0.8029, 0.5738, 0.1409, 0.0395)	−3.3220	210	

condition of the algorithm for terminating. Figure 2 presents the main steps of iCOA as a pseudo-code.

The only difference between the main COA and the proposed method is in step 3. The mentioned step computes the value of the α by the β coefficient. The new computed α will be used for defining ELR in step 4.

3 Experiment and results

In this section, seven commonly explored problems are used to demonstrate the performance of the iCOA and the obtained results are compared with COA and other known methods that use the same problems. The proposed method is coded in MATLAB programming software and simulations have run on a Pentium IV 2.8 GHz with 1.5 GB RAM. The best results of benchmarks are obtained by 30 independent runs. The

Table 6 Results of iCOA (present study)

Ordinal number	Optimal solution $(x_1, x_2, x_3, x_4, x_5, x_6)$	Optimal value	Iteration iterations	Mean iteration of 10 runs
1	(0.2028, 0.1473, 0.4769, 0.2745, 0.3118, 0.6572)	−3.3220	50	63.3
2	(0.2028, 0.1465, 0.4767, 0.2749, 0.3118, 0.6575)	−3.3220	33	
3	(0.2015, 0.1451, 0.4766, 0.2754, 0.3118, 0.6572)	−3.3220	117	
4	(0.2009, 0.1473, 0.4778, 0.2754, 0.3121, 0.6578)	−3.3220	48	
5	(0.2004, 0.1467, 0.4777, 0.2748, 0.3118, 0.6567)	−3.3220	80	
6	(0.2004, 0.1462, 0.4764, 0.2752, 0.3120, 0.6577)	−3.3220	67	
7	(0.2009, 0.1478, 0.4757, 0.2757, 0.3118, 0.6576)	−3.3220	64	
8	(0.2009, 0.1466, 0.4758, 0.2749, 0.3121, 0.6568)	−3.3220	78	
9	(0.2007, 0.1467, 0.4774, 0.2755, 0.3118, 0.6571)	−3.3220	55	
10	(0.2028, 0.1474, 0.4762, 0.2754, 0.3112, 0.6576)	−3.3220	41	

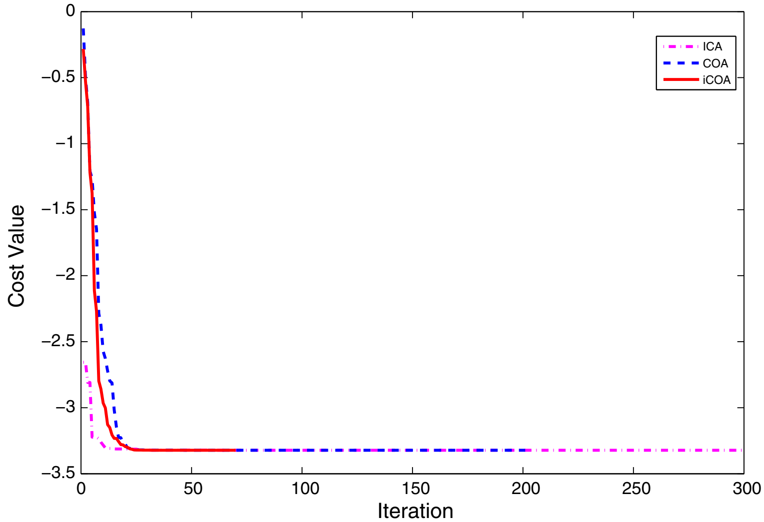

Fig. 3 The convergence history of Test 1

used parameters for solving the problems are shown in Table 2 and the suitable values for the parameters of each benchmark are accessible by a few tests using different values. Since the size of population in each iteration of the iCOA is different, so, the number of function evaluations (NFEs) is the standard of the iCOA in comparison with the prior methods (See Tables 16, 17, 18).

Test 1: The Hartman's function [11]

$f(x) = -\sum_{i=1}^{4} c_i exp\left[-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right]$, where $0 \le x_j \le 1$, $c = (1\ 1.2\ 3\ 3.2)$,

Table 7 Results of Test 2 (from [12])

Variables	Initial iteration	After 100 iterations	After 200 iterations	After 300 iterations	After 400 iterations	After 500 iterations
x_1	4.9203	5.3737	5.3667	5.3656	5.3626	5.3623
x_2	4.6815	5.3564	5.3601	5.3618	5.3628	5.3624
x_3	4.9207	5.3522	5.3651	5.3658	5.3627	5.3621
x_4	5.5048	5.3846	5.3656	5.3648	5.3636	5.3633
x_5	6.3685	5.3597	5.3628	5.3630	5.3607	5.3627
x_6	6.7112	5.3520	5.3669	5.3640	5.3631	5.3625
x_7	5.6790	5.3369	5.3621	5.3626	5.3613	5.3624
x_8	6.3557	5.3420	5.3626	5.3629	5.3623	5.3622
x_9	11.7889	5.3705	5.3574	5.3647	5.3627	5.3627
x_{10}	10.3531	5.3515	5.3580	5.3592	5.3622	5.3616
$f(x)$	−7.690599	−12.158781	−12.159769	−12.159797	−12.15981	−12.15982

Table 8 The results of ICA for Test 2 with $D = 10$ (from [19])

Variables	Initial iteration	After 100 iterations	After 200	After 300 iterations	After 400 iterations	After 500 iterations
x_1	7.648336	5.362271	5.362271	5.362271	5.362271	5.362271
x_2	6.285073	5.362749	5.362749	5.362749	5.362749	5.362749
x_3	6.441411	5.362276	5.362276	5.362276	5.362276	5.362276
x_4	5.521101	5.362543	5.362543	5.362543	5.362543	5.362543
x_5	6.255337	5.363662	5.363662	5.363662	5.363662	5.363662
x_6	9.860261	5.362470	5.362470	5.362470	5.362470	5.362470
x_7	9.630791	5.362061	5.362061	5.362061	5.362061	5.362061
x_8	4.882258	5.362417	5.362417	5.362417	5.362417	5.362417
x_9	5.152085	5.363256	5.363256	5.363256	5.363256	5.363256
x_{10}	5.451308	5.361964	5.361964	5.361964	5.361964	5.361964
$f(x)$	−7.36443399	−12.15982	−12.15982	−12.15982	−12.15982	−12.15982

$$\mathbf{p_{ij}} = \begin{bmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1415 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{bmatrix},$$

$$\mathbf{a_{ij}} = \begin{bmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix}$$

where $\min f(x) = -3.3220$. The iCOA and COA were run 10 times such as Mo et al. [11] and ICA [19]. The results of iCOA with mean of 1,693 NFEs, the COA

Table 9 The results of COA for Test 2 with $D = 10$

Variables	Initial iteration	After 100 iterations	After 200 iterations	After 300 iterations	After 400 iterations	After 500 iterations
x_1	6.392383	5.337477	5.363541	5.362890	5.362890	5.362890
x_2	3.862854	5.315841	5.363531	5.363373	5.363373	5.363373
x_3	9.362560	5.345075	5.361098	5.362124	5.362124	5.362124
x_4	5.077271	5.466212	5.361254	5.362350	5.362350	5.362350
x_5	5.996354	5.367775	5.363504	5.362215	5.362215	5.362215
x_6	6.243231	5.333418	5.361857	5.362189	5.362189	5.362189
x_7	6.688654	5.378127	5.360857	5.362560	5.362560	5.362560
x_8	4.879359	5.386594	5.362075	5.362725	5.362725	5.362725
x_9	11.458427	5.421125	5.363379	5.362194	5.362194	5.362194
x_{10}	7.759052	5.351978	5.362788	5.362144	5.362144	5.362144
$f(x)$	−5.48872713	−12.150484	−12.159816	−**12.15982**	−12.15982	−12.15982

The best solutions obtained by the proposed COA and iCOA are in bold

Table 10 The results of iCOA for Test 2 with $D = 10$ (present study)

Variables	Initial iteration	After 100 iterations	After 200 iterations	After 300 iterations	After 400 iterations	After 500 iterations
x_1	5.684124	5.363014	5.363014	5.363014	5.363014	5.363014
x_2	7.860259	5.362196	5.362196	5.362196	5.362196	5.362196
x_3	9.925422	5.360975	5.360975	5.360975	5.360975	5.360975
x_4	5.835342	5.361543	5.361543	5.361543	5.361543	5.361543
x_5	6.056260	5.362353	5.362353	5.362353	5.362353	5.362353
x_6	4.762997	5.361632	5.361632	5.361632	5.361632	5.361632
x_7	8.836812	5.362674	5.362674	5.362674	5.362674	5.362674
x_8	4.373674	5.361906	5.361906	5.361906	5.361906	5.361906
x_9	9.387655	5.361970	5.361970	5.361970	5.361970	5.361970
x_{10}	11.726187	5.361894	5.361894	5.361894	5.361894	5.361894
$f(x)$	4.61523058	−**12.15982**	−12.15982	−12.15982	−12.15982	−12.15982

The best solutions obtained by the proposed COA and iCOA are in bold

Table 11 Comparison results of Test 2 with $D = 100$

f(x)	Initial iteration	After 1000 iterations	After 2000 iterations	After 3000 iterations	After 4000 iterations	After 5000 iterations	After 6000 iterations
PPSO [12]	54.103342	121.208321	121.554754	121.593659	121.596941	121.598050	121.598204
ICA [19]	29.786871	121.598200	121.598200	121.598200	121.598200	121.598200	121.598200
COA	17.647195	116.553416	117.789701	121.107134	**121.5982**	121.5982	121.5982
iCOA	24.195038	**121.598200**	121.598200	121.598200	121.598200	121.598200	121.598200

The best solutions obtained by the proposed COA and iCOA are in bold

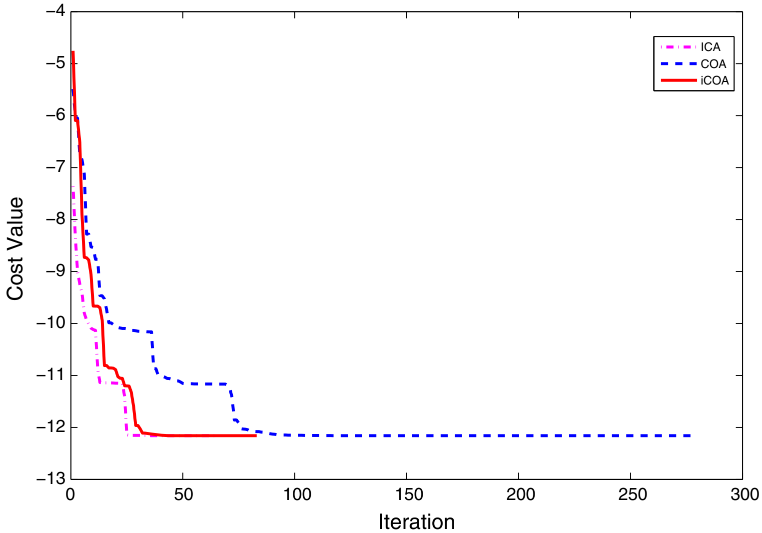

Fig. 4 The convergence history of Test 2 with $D = 10$

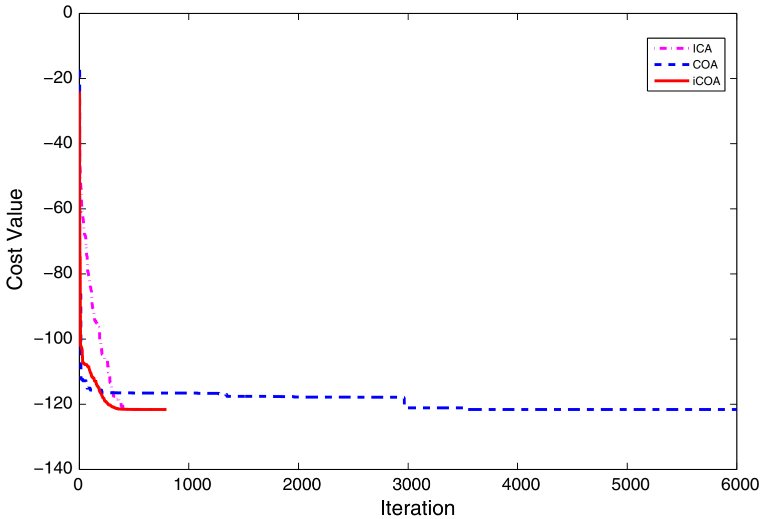

Fig. 5 The convergence history of Test 2 with $D = 100$

with mean of 3961 NFEs, Mo et al. [11] and ICA [19] with 300 iterations and 300 population (90,000 NFEs) are shown in Tables 3, 4, 5, 6 respectively. Figure 3 shows the convergence chart of ICA, COA and iCOA for Test 1.

Test 2: This example was given in [12] and [19]

$$\min f(x) = \sum_{i=1}^{D} \left[\sin(x_i) + \sin\left(\frac{2x_i}{3}\right) \right]$$

The solution of this function is 1.21598D and the variables are in (3, 13).

The results of PPSO [12] with 1000 iterations and 300 population (300,000 NFEs) for $D = 10$ and 6000 iterations and 300 population (1,800,000 NFEs) for $D = 100$, ICA [19] with 70 iterations and 300 population (21,000 NFEs) for $D = 10$ and 700 iterations and 300 population (210,000 NFEs) for $D = 100$ and the COA with the mean of 32,114 NFES for $D = 10$ and 382,180 NFEs for $D = 100$ and our algorithm with the mean of 15,130 NFES for $D = 10$ and 57,995 NFEs for $D = 100$ are comparable in Tables 7, 8, 9, 10, 11 respectively. The iCOA solved Test 2 quicker than the prior methods with less number of function evaluations

(See Figs. 4 and 5 too).

4 Case study

In this section, five standard systems are selected from the literatures to demonstrate the efficiency of the iCOA for solving systems of nonlinear equations.

Case 1

$$\begin{aligned}
&x_1^{x_2} + x_2^{x_1} - 5x_1x_2x_3 = 85\\
&x_1^3 - x_2^{x_3} - x_3^{x_2} = 60\\
&x_1^{x_3} + x_3^{x_1} - x_2 = 2\\
&3 \le x_1 \le 5, \quad 2 \le x_2 \le 4, \quad 0.5 \le x_3 \le 2.
\end{aligned}$$

The solution in [11,12] and [19] was (4, 3, 1). The iCOA method got the same result but the convergence history of iCOA is better with 200 iterations and the mean of 9869 NFEs whereas [12,19] and COA had been reached to the answer with 1000 iterations

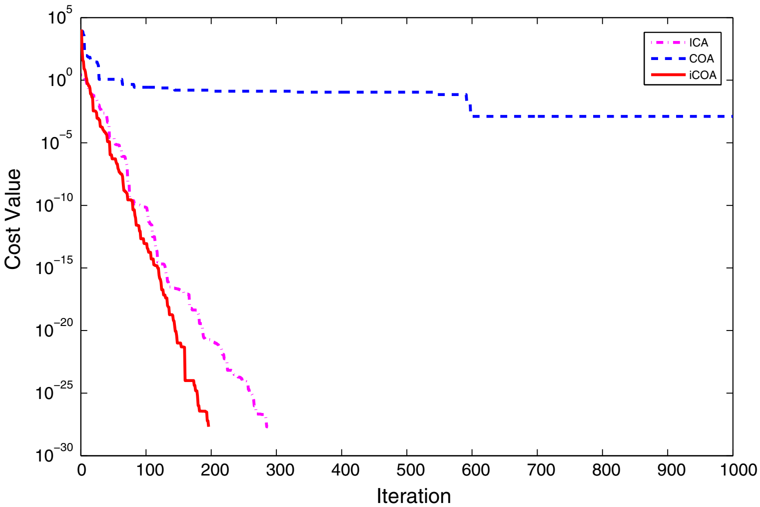

Fig. 6 The convergence history of Case 1

Table 12 Comparison results of iCOA for Case 2 with [12,19,20]

Methods	x_1	x_2	$f(x)$
PPSO [12] and Gyurhan [20]	−0.29051455550725	1.08421508149135	4.686326815078573e−029
PPSO [12] and Gyurhan [20]	−0.793700525984100	−0.793700525984100	1.577721810442024e−030
ICA [19]	1.084215081491351	−0.290514555507251	3.562200025138631e−030
ICA [19]	−0.793700525984100	−0.793700525984100	1.577721810442024e−030
ICA [19]	−0.290514555507251	1.084215081491351	3.562200025138631e−030
COA	1.084215081563733	−0.290514554550255	1.315912686315398e−017
COA	−0.793700525199129	−0.793700527168166	2.883311723428020e−017
COA	−0.290514556862374	1.084215081072923	2.873662884066830e−017
iCOA (present study)	1.0842150814913512220446049250313	−0.2905145555072514440892098500626	0
iCOA (present study)	−0.793700525984100	−0.793700525984100	1.972152263052530e−031
iCOA (present study)	−0.2905145555072514440892098500626	1.0842150814913512220446049250313	0

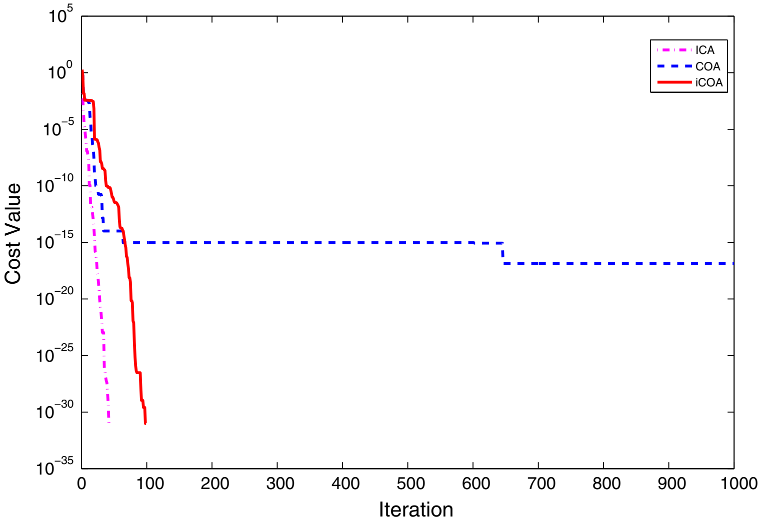

Fig. 7 The convergence history of Case 2

and 250 population (250,000 NFEs), 300 iterations and 250 population (75,000 NFEs) and the mean of 38,426 NFEs respectively. See Fig. 6.

Case 2 This example was given in [12,19,20]

$$
\begin{aligned}
x_1^3 - 3x_1x_2^2 - 1 = 0 \\
3x_1^2x_2 - x_2^3 + 1 = 0
\end{aligned}
$$

The solutions in [12] and [20] were obtained with 120 iterations and an unknown number of population. The parameters of the ICA [19] method are 50 iterations with 250 countries (12,500 NFEs) and the mean NFEs of COA is 24,153 NFEs while the iCOA method got the better and more accurate results with 3,659 NFEs (see Table 12). Figure 7 shows the convergence history of Case 2.

Case 3 Neurophysiology application (benchmark in [13])

$$
\begin{aligned}
& x_1^2 + x_3^2 = 1 \\
& x_2^2 + x_4^2 = 1 \\
& x_5x_3^3 + x_6x_4^3 = 0 \\
& x_5x_1^3 + x_6x_2^3 = 0 \\
& x_5x_1x_3^2 + x_6x_4^2x_2 = 0 \\
& x_5x_1^2x_3 + x_6x_2^2x_4 = 0 \\
& -10 \le x_i \le 10, \quad 1 \le i \le 6
\end{aligned}
$$

We considered the example proposed in [10,21] and [19] with 200 iterations and 300 population (60,000 NFEs). The best known solutions of [10,13,19,21] and COA

Table 13 Comparison results of Case 3

Method	Variables values ($x_1, \ldots, x_6$)	Functions values ($f_1, \ldots, f_6$)
The best results of [10]	−0.8078668904	0.0050092197
	−0.9560562726	0.0366973076
	0.5850998782	0.0124852708
	−0.2219439027	0.0276342907
	0.0620152964	0.0168784849
	−0.0057942792	0.0248569233
The best results of ICA [19]	−0.041096050919063	0
	0.041096050919063	0
	0.999155200456294	0
	−0.999155200456294	0
	0.098733550533454	0
	0.098733550533454	0
The best results of fuzzy ASA [13]	−0.03810884298576576	0
	−0.03810884298576576	0
	0.9992735942104576	0
	0.9992735942104576	0
	−0.3554963090941105	1.734723475976807e−018
	0.3554963090941105	−1.084202172485504e−019
The best of COA	−0.000024724870196	6.113192174694859e−010
	−0.000024931668652	6.215881143134538e−010
	1.000000000000000	0
	1.000000000000000	3.823436746661753e−016
	0.999746640562583	2.067460616548185e−007
	−0.999746640562583	−1.026629384236605e−011
The best of iCOA	−0.021696669339335	0
	−0.021696669339335	0
	0.999764599563107	0
	0.999764599563107	0
	−0.999924486393478	0
	0.999924486393478	0

have been shown in Table 13 beside the exact solution of iCOA with the mean of 16,010 NFEs.

The convergence history of ICA [19], COA and iCOA for Case 3 are shown in Fig. 8.

Case 4 (Problem 2 in [22], Test Problem 14.1.4 in [23], Case study in [19])

$$f_1(x_1, x_2) = 0.5 \sin(x_1 x_2) - 0.25 x_2/\pi - 0.5 x_1 = 0$$

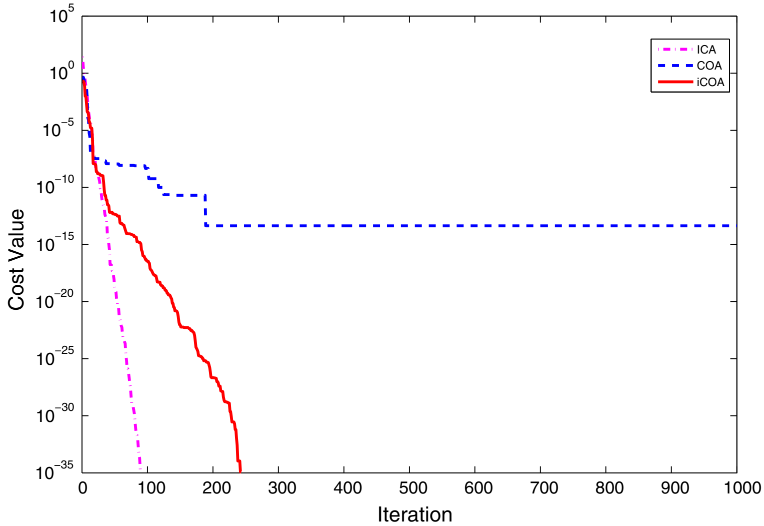

Fig. 8 The convergence history of Case 3

$$f_2(x_1, x_2) = (1 - 0.25/\pi)(\exp(2x_1) - e) + ex_2/\pi - 2ex_1 = 0$$
$$0.25 \leq x_1 \leq 1, \quad 1.5 \leq x_2 \leq 2\pi.$$

The known solution for the auxiliary function of Case 4 in [22] is 7.693745216994-211e−008 and the best solutions of Case 4 in [23] are (0.29945, 2.83693) and (0.5, 3.14159).

The results of the ICA [19] with 250 iterations and 250 countries (means 65,500 NFEs) the same as [22] are 5.631272867601562e−024 and 0. The best solution of COA with the mean of 46,713 NFEs is 6.182101709455280e−014 while the results of iCOA with the mean of 10,012 NFEs are 7.703719777548943e−034 and 0.

The iCOA has found better results than the mentioned methods with less number of function evaluations. The comparison of obtained results are accessible in Table 14 (see Fig. 9 too).

Case 5 (Problem 6 in [22] and Test Problem 14.1.6 in [23])

This problem has been solved by the filled function method in [22], and proposed the problem in [23] and [19].

$$\begin{aligned}
&4.731 \times 10^{-3}x_1x_3 - 0.3578x_2x_3 - 0.1238x_1 + x_7 - 1.637\\
&\times 10^{-3}x_2 - 0.9338x_4 - 0.3571 = 0\\
&0.2238x_1x_3 + 0.7623x_2x_3 + 0.2638x_1 - x_7 - 0.07745x_2 - 0.6734x_4 - 0.6022 = 0\\
&x_6x_8 + 0.3578x_1 + 4.731 \times 10^{-3}x_2 = 0\\
&-0.7623x_1 + 0.2238x_2 + 0.3461 = 0\\
&x_1^2 + x_2^2 - 1 = 0\\
&x_3^2 + x_4^2 - 1 = 0
\end{aligned}$$

Table 14 Comparison results of iCOA for Case 4 with [19,22,23] and COA

Methods	x_1	x_2	f_1	f_2	$f(x)$
The best in [22]	0.50043285	3.14186317	−0.00023852	0.00014159	7.693745216994211e−008
The best in [23]	0.29945	2.83693	NA	NA	NA
The best in [23]	0.5	3.14159	NA	NA	NA
The best of ICA [19]	0.299448692495720	2.836927770471037	1.305289210051797e-012	2.284838984678572e−013	5.631272867601562e−024
The best of ICA [19]	0.500000000000000	3.141592653589794	0	0	0
The best of COA	0.500000226586182	3.141593003459494	−1.411350338820228e − 007	2.046995830617959e−007	6.182101709455280e-014
The best of iCOA	0.299448692490926	2.836927770458940	2.775557561562891e−017	0	7.703719777548943e−034
The best of iCOA	0.500000000000000	3.141592653589794	0	0	0

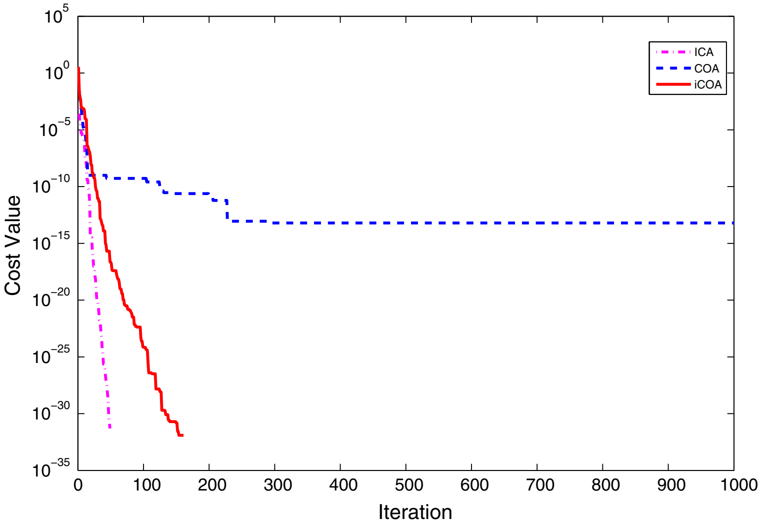

Fig. 9 The convergence history of Case 4

$$x_5^2 + x_6^2 - 1 = 0$$
$$x_7^2 + x_8^2 - 1 = 0$$
$$-1 \leq x_i \leq 1, \quad i = 1, \ldots, 8.$$

The most known solution of Case 5 in [19,22,23] with 1000 iterations and 300 population (300,000 NFEs) and COA with 153,671 NFEs and our results with the mean of 27,203 NFEs are shown in Table 15. The convergence history of ICA [19], COA and iCOA are shown in Fig. 10.

4.1 Discussion

There are a number of different mathematical and evolutionary methods for solving the systems of nonlinear equations. In this paper, our best solutions are compared with other methods such as the Hybrid Approach with Chaos Optimization and Quasi–Newton, the Conjugate Direction Particle Swarm Optimization, the Proposed Particle Swarm Optimization, GA, a New Filled Function Method, the Homotopies Exploiting Newton Polytopes, Imperialist Competitive Algorithm and Cuckoo Optimization Algorithm. Comparison of the all obtained results indicates that the iCOA outperforms the mentioned methods with much less number of function evaluations. For example, we reached the exact solution of Case 4 with the mean of 10,012 function evaluations and 25 iterations in comparison with the COA with the mean of 46,713 function evaluations and the ICA with 120,000 function evaluations and 400 iterations. According to Tables 12 and 14, the iCOA finds solutions more accurately than the mentioned methods. See Table 11 for a larger scale problem at which the iCOA runs better than the previously printed methods.

Table 15 Comparison results of Case 5

Method	Variables values ($x_1, \ldots, x_8$)	Functions values ($f_1, \ldots, f_8$)
The best in [22]	0.67154465	−0.00000375
	0.74097111	0.00001537
	0.95189459	0.00000899
	−0.30643725	0.00001084
	0.96381470	0.00001039
	−0.26657405	0.00000709
	0.40463693	0.00000049
	0.91447470	−0.00000498
The best in [23]	0.1644	−8.8531e−005
	−0.9864	3.5894e−005
	−0.9471	6.6216e−006
	−0.3210	2.1560e−005
	−0.9982	1.2320e−005
	−0.0594	3.9410e−005
	0.4110	−6.8400e−005
	0.9116	−6.4440e−005
The best of ICA [19]	0.164431665854327	2.775557561562891e−016
	−0.986388476850967	−1.110223024625157e−016
	0.718452601027603	1.734723475976807e−018
	−0.695575919707312	1.665334536937735e−016
	0.997964383970433	0
	0.063773727557003	0
	−0.527809105283546	0
	−0.849363025083964	0
The best of COA	0.164143039609694	9.986069578277190e−005
	−0.986427720777058	−7.692972597672654e−005
	0.718379409996089	1.204919703285861e−004
	−0.695633517102380	2.112369956246973e−004
	−0.997944511392644	−1.741423026913047e−005
	0.063504300944208	−2.503317744295686e−005
	−0.527782043140892	−7.395594288472918e−005
	−0.849440073396218	1.023233533463674e−004
The best of iCOA	0.671554261818887	0
	0.740955378840649	0
	0.951892748840980	−1.695692197767329e−016
	−0.306431386616911	5.551115123125783e−017
	−0.963810765487133	0
	−0.266587337154461	0
	0.404641388921954	0
	0.914475448752623	0

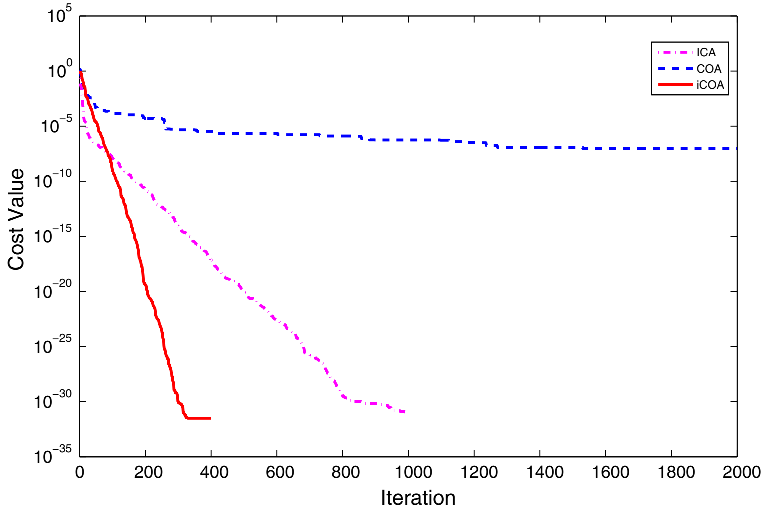

Fig. 10 The convergence history of Case 5

Table 16 Statistical results of NFEs for tests and cases

Problem	N	Mean of NFEs		Maximum of NFEs		Minimum of NFEs	
		COA	iCOA	COA	iCOA	COA	iCOA
Test 1	30	3961	1693	10744	2013	1459	1450
Test 2 ($D = 10$)	30	32,114	15,130	45,577	18,311	22,015	11,494
Test 2 ($D = 100$)	30	382,180	57,995	428,944	59,128	21,140	56,658
Case 1	30	38,426	9869	48,858	10,365	13,275	9507
Case 2	30	24,153	3659	34,055	3960	6030	3304
Case 3	30	50,223	16,010	63,540	17,144	25,097	15,534
Case 4	30	46,713	10,012	62,191	10,521	18,366	9349
Case 5	30	15,3671	27,203	165,396	27,836	116,496	23,962

The efficiency of the COA increases by applying the corrected ELR to it. This strategy improves the performance of the iCOA significantly. The statistical results of the benchmarks with 30 independent runs in Tables 16, 17, 18 indicate the stability and convergence of our proposed algorithm are reliable.

The comparison of convergence history of ICA, COA and iCOA methods are presented in Figs. 3, 4, 5, 6, 7, 8, 9, 10. The quality of the obtained solutions by iCOA in all of the tests and case studies is better. It is notable that the comparison is based on the number of fitness function evaluations and the number of iteration in the charts is not important. So the priority of the convergence history of the methods in the Figs. 7, 8, 9 are not considerable.

Table 17 The statistical results of COA

Problem	N	Mean	SD	SE mean	Worst	Best
Test 1	30	−3.270236666666667	6.0207597660860e−002	1.0992353124016e−002	−3.200300000000000	−3.322000000000000
Test 2 ($D = 10$)	30	12.159820173609727	1.749886958312007e−007	3.194841867171962e−008	12.159820745007874	12.159820007423241
Test 2 ($D = 100$)	30	118.7731522763454	1.313101848178492	2.43836891758417e−001	117.6067613334681	121.5981841573864
Case 1	30	3.4324984020375e−002	4.5836466960008e−002	8.368555636779e−003	1.72299626076878e−001	1.277641345977e−003
Case 2	30	1.162170793603723e−015	2.656193789782612e−015	4.849524185896906e−016	1.441120398450452e−014	1.315912686315398e−017
Case 3	30	7.864894633799755e−011	1.139168992800978e−010	2.079828513891786e−011	3.785467246705225e−010	4.274469419814430e−014
Case 4	30	1.136979107153005e−011	2.452570695165942e−011	4.477760978728376e−012	9.965352981807610e−011	6.182101709455280e−014
Case 5	30	2.1078049688554e−002	6.4312631257478e−002	1.1741826290744e−002	2.10800494613618e−001	9.189919005205856e−008

Table 18 The statistical results of iCOA

Problem	N	Mean	SD	SE mean	Worst	Best
Test 1	30	−3.322000000000002	1.806724113306785e−015	3.298611840155485e−016	−3.322000000000000	−3.322000000000000
Test 2 ($D = 10$)	30	12.159820258933882	2.575040622155351e−007	4.701359450822076e−008	12.159820865087692	12.159819983047303
Test 2 ($D = 100$)	30	121.5982004232355	3.557301962365390e−007	6.494715095483059e−008	121.5982013467766	121.5982000101087
Case 1	30	0.0	0.0	0.0	0.0	0.0
Case 2	30	3.040401405539317e−032	1.022966074287776e−031	1.867671981499734e−032	4.930380657631324e−031	0.0
Case 3	30	9.417141153384007e−032	2.717456541168603e−031	4.961374155460034e−032	1.281204196196034e−030	0.0
Case 4	30	1.679410911505670e−031	4.278776623313279e−031	7.811941583714892e−032	1.751055505436875e−030	0.0
Case 5	30	3.120368750246472e−031	3.047901998633619e−031	5.564682259055711e−032	9.737501798821865e−031	3.183520820670952e−032

5 Conclusions and future works

An improved evolutionary algorithm based on the basic version of the COA is proposed. It eliminates the disadvantage of COA by changing the policy of ELR. Some well-known problems are presented to demonstrate the efficiency of the iCOA in comparison with other algorithms such as the COA, ICA, PPSO, Conjugate Direction Particle Swarm Optimization, GA, Filled Function Method and the Homotopies Exploiting Newton Polytopes. Furthermore, the mentioned improved algorithm finds solutions for problems with the lowest number of function evaluations which helps to increase the speed of finding answers. Additionally, with the corrected ELR, iCOA found more accurate solutions than any other method. As a part of our future work, the convergence speed of the iCOA can be raised by the use of chaos theory for φ [24]. Furthermore, we are planning to extend the iCOA on solving the constrained nonlinear problems.

References

1. El-Emary IMM, Abd El-Kareem MM (2008) Toward using genetic algorithm for solving nonlinear eqution systems. World Appl Sci J 5:282–289
2. Zhijian W, Kang L (2003) A fast and elitist parallel evolutionary algorithm for solving systems of non-linear equations. In: The 2003 Congress on Evol Comput vol 2, pp 1026–1028
3. Mastorakis NE (2005) Solving non-linear equations via genetic algorithms. In: Proceedings of the 6th WSEAS international conference on evoluationary computing vol 6, pp 24–28
4. Li G, Zeng Z (2008) A neural-network algorithm for solving nonlinear equation systems. In: IEEE international conference on computational intelligence and security, vol 1, pp 20–23
5. Huan-Tong G, Yi-Jie S, Qing-Xi S, Ting-Ting W (2009) Research of ranking method in evolution strategy for solving nonlinear system of equations. In: 1st international conference on information science and engineering, vol 1, 348–351
6. Abdollahi M, Isazadeh A, Abdollahi D (2013) Solving the constrained nonlinear optimization based on imperialist competitive algorithm. Int J Nonlinear Sci 15:212–219
7. Ouyang A, Zhou Y, Luo Q (2009) Hybrid particle swarm optimization algorithm for solving systems of nonlinear equations. In: IEEE international conference on granular computing, GRC09, pp 460–465
8. Wu J, Cui Z, Liu J (2011) Using hybrid social emotional optimization algorithm with metropolis rule to solve nonlinear equations. In: 10th IEEE international conference on cognitive informatics and cognitive computing, vol 10, pp 405–411
9. Luo YZ, Tang GJ, Zhou LN (2008) Hybrid approach for solving systems of nonlinear equations using chaos optimization and quasi-Newton method. Appl Soft Comput 8:1068–1073
10. Grosan C, Abraham A (2008) A new approach for solving nonlinear equations systems. IEEE Trans Syst Man Cybern Part A Syst Hum 38:698–714
11. Mo Y, Liu H, Wang Q (2009) Conjugate direction particle swarm optimization solving systems of nonlinear equations. Comput Math Appl 57:1877–1882
12. Jaberipour M, Khorram E, Karimi B (2011) Particle swarm algorithm for solving systems of nonlinear equations. Comput Math Appl 62:566–576
13. Oliveira HA Jr, Petraglia A (2013) Solving nonlinear systems of functional equations with fuzzy adaptivesimulated annealing. Appl Soft Comput 13:4349–4357
14. Henderson N, Sacco WF, Platt GM (2010) Finding more than one root of nonlinear equations via a polarization technique: an application to double retrograde vaporization. Chem Eng Res Des 88:551–561
15. Pourjafari E, Mojallali H (2012) Solving nonlinear equations systems with a new approach based on invasive weed optimization algorithm and clustering. Swarm Evol Comput 4:33–43
16. Kuri-Morales AF (2003) Solution of simultaneous non-linear quations using genetic algorithms. WSEAS Trans Syst 2:44–51

17. Mastorakis NE (2005) solving non-linear equations via genetic algorithms. WSEAS Evolut Comput Lisbon Portugal 6:24–28
18. Rajabioun R (2011) Cuckoo optimization algorithm. Appl Soft Comput 11:5508–5518
19. Abdollahi M, Isazadeh A, Abdollahi D (2013) Imperialist competitive algorithm for solving systems of nonlinear equations. Elsevier Comput Math Appl 65:1894–1908
20. Nedzhibov GH (2008) A family of multi-point iterative methods for solving systems of nonlinear equations. Comput Appl Math 222:244–250
21. Verschelde J, Verlinden P, Cools R (1994) Homotopies exploiting Newton polytopes for solving sparse polynomial systems. SIAM J Numer Anal 31:915–930
22. Wang C, Luo R, Wu K, Han B (2011) A new filled function method for an unconstrained nonlinear equation. Comput Appl Math 235:1689–1699
23. Floudas CA, Pardalos PM, Adjiman CS, Esposito WR, Gumus ZH, Harding ST, Klepeis JL, Meyer CA, Schweiger CA (1999) Handbook of test problems in local and global optimization. Kluwer Academic Publishers, Dordrecht
24. Bahrami H, Faez K, Abdechiri M (2010) Imperialistic competitive algorithm using chaos theory for optimization. In: 12th international conference on computer modeling and stimulation, vol 12, pp 98–103