Appl Intell (2018) 48:2681–2697
https://doi.org/10.1007/s10489-017-1096-8

Improved cat swarm optimization algorithm for solving global optimization problems and its application to clustering

Yugal Kumar[1] · **Pradeep Kumar Singh**[1]

Published online: 19 December 2017

Abstract This paper presents a cat swarm optimization (CSO) algorithm for solving global optimization problems. In CSO algorithm, some modifications are incorporated to improve its performance and balance between global and local search. In tracing mode of the CSO algorithm, a new search equation is proposed to guide the search toward a global optimal solution. A local search method is incorporated to improve the quality of solution and overcome the local optima problem. The proposed algorithm is named as Improved CSO (ICSO) and the performance of the ICSO algorithm is tested on twelve benchmark test functions. These test functions are widely used to evaluate the performance of new optimization algorithms. The experimental results confirm that the proposed algorithm gives better results than the other algorithms. In addition, the proposed ICSO algorithm is also applied for solving the clustering problems. The performance of the ICSO algorithm is evaluated on five datasets taken from the UCI repository. The simulation results show that ICSO-based clustering algorithm gives better performance than other existing clustering algorithms.

Keywords Cat swarm optimization · Clustering · Meta-heuristics · Numerical functions · Improved CSO

✉ Yugal Kumar
yugalkumar.14@gmail.com

Pradeep Kumar Singh
pradeep_84cs@yahoo.com

[1] Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat, Solan, Himachal Pradesh, India

1 Introduction

There are many optimization problems, including unimodal and multimodal, in the fields of engineering and science. These optimization problems can be further categorized into unimodal separable and inseparable problems, and multimodal separable and inseparable problems. In literature, many methods have been reported for solving these problems; these methods either maximize or minimize the objective function. Moreover, these methods have also been applied for solving real-life problems such as clustering, classification, scheduling, path planning, resource allocation, and many other problems. In recent years, large numbers of optimization algorithms have been developed by researchers to find the solution for the above-mentioned optimization problems. These algorithms are divided into two categories, i.e., exact algorithms and approximation algorithms [1]. Exact algorithms find the optimal solution within the bounded time but have exponential computational time, whereas approximation algorithms provide better results both in time and solution using heuristics. Meta-heuristic algorithms are the sub-branch of approximation algorithms which have been applied for solving a wide range of optimization problems. Most of these algorithms are inspired by natural phenomena. Some examples of these algorithms are simulated annealing (SA) algorithm [2], genetic algorithm (GA) [3], particle swarm optimization (PSO) [4], ant colony optimization (ACO) [5], harmony search (HS) [6], artificial bee colony (ABC) [7], firefly algorithm (FA) [8], league championship algorithm (LCA) [9], water cycle algorithm (WCA) [10], charge system search (CSS) algorithm [11, 12], magnetic charge system search (MCSS) [13–15], teaching learning based

optimization (TLBO) [16, 17], and mine blast algorithm (MBA) [18].

Recently, the CSO algorithm has gained popularity among the scientific community due to its strong exploration ability and effective solving of complex optimization problems [19]. Due to its simple and straightforward implementation, this algorithm has been applied in diverse fields. Initially, Chu et al. presented the CSO algorithm for optimizing numerical test functions [19]. This algorithm is characterized by the behavior of cats, and each cat is described in terms of position and velocity. This algorithm measures the behavior of a cat in two modes – seeking mode and tracing mode. In the optimization process, the position of a cat is optimized by using its own experience (especially, in seeking mode) and velocity. However in tracing mode, the CSO algorithm can trap in local optima for some complex and intricate optimization problems, and also can have low convergence rate in last iterations. The CSO algorithm has weak diversity problem for a few benchmark functions.

Contribution of this work The main contribution of this study is that it handles the aforementioned shortcomings of the CSO algorithm. The aim of this research work is to propose some modifications in the CSO algorithm in order to overcome the problems of weak diversity and convergence rate. To make the CSO algorithm more efficient, reliable and robust, these modifications are incorporated into individual cat memory and experience. The main contribution of this work is to deal with the following issues of CSO algorithm.

- Lack of balance between exploration (seeking mode) and exploitation (tracing mode) processes.
- Lack of diversity for fewer benchmark functions, especially in tracing mode.
- Slow convergence problem in the last generations of the iterative process.

The following amendments are integrated in the CSO algorithm to deal with the above-mentioned issues. These amendments are as follows.

- A new enhanced and accelerated velocity equation is proposed for the CSO algorithm, especially for tracing mode.
- A new position update equation is proposed both for tracing and seeking modes.
- A local search method is incorporated to improve the quality of solution and handle local the optima problem.

The proposed algorithm is tested on several benchmark test functions for analyzing its performance and accuracy. The experimental results showed that the performance of the proposed algorithm is improved in terms of global search and convergence rate.

The rest of the paper is organized as follows. Section 2 summarizes related works on CSO algorithm. The description of CSO algorithm is illustrated in Section 3. Section 4 presents the proposed ICSO algorithm and its steps for solving benchmark test functions. The experimental results of the proposed algorithm and other state-of-the-art algorithms are given in Section 5. Section 6 explores the applicability of the proposed ICSO algorithm for solving the real-world clustering problems. Section 7 reports the conclusion of the work done.

2 Related works

This section describes the related works in the direction of CSO algorithm for solving different optimization problems. Mohapatra et al. [20] developed a modified cat swarm optimization algorithm for improving the searching ability of cats. In this work, a mutation operator was applied to mutate the position of the best cat for achieving good results. The gene data analysis and classification tests showed that the modified CSO algorithm has better performance in comparison to other algorithms. To investigate the parallel structure of CSO algorithm, Tasi et al. [21] presented a parallel CSO algorithm and tested its performance on several benchmark functions. It was reported that the parallel CSO algorithm has better performance to PSO and CSO algorithms. In continuation of their work, Tasi et al. [22] developed an enhanced parallel cat swarm optimization (EPSO) algorithm to obtain higher accuracy and less computational time. In the EPSO algorithm, the Taguchi method was incorporated for achieving higher accuracy rate and optimal solutions. The performance of the EPSO algorithm was tested on five benchmark functions, and it was found that the EPSO algorithm has a higher accuracy rate. Orouskhani et al. introduced the concept of inertia weight in CSO algorithm for better convergence rate. The simulation results showed that the proposed algorithm achieves better convergence results in comparison to the original CSO algorithm [23]. For optimal design of the circular and concentric circular antenna arrays, Ram et al. [24] applied the CSO algorithm. Yang et al. [25] adopted the CSO algorithm for processing and analyzing the medical image. In their work, CSO algorithm was combined with limited memory Broyden–Fletcher–Goldfarb–Shanno with boundaries (L-BFGS-B) and called it HLCSO. The simulation results showed that the HLCSO method gives better accuracy and less computational times. To improve the performance of the CSO algorithm, Lin et al.

[26] incorporated two improvements in CSO algorithm and called the resultant algorithm an improved CSO (ICSO). Results stated that the ICSO algorithm has higher accuracy rate in comparison to the traditional CSO algorithm. Guo et al. [27] applied the CSO algorithm to estimate the parameters of solar cell model. Simulation results showed that the CSO is an effective tool for identification of parameters of solar cell model. To optimize the design of a channel cross-section, Liu et al. [28] applied the CSO algorithm for safety and stability of side walls in a hydraulic design. The CSO algorithm provides better results in comparison to GA and PSO algorithms. In continuation of their work, Ram et al. [29] also applied CSO algorithm for reducing the side lobe level of the concentric circular antenna and satisfactory results. To determine the location and controlling parameters of SVC and TCSC, Nireekshana et al. [30] applied CSO algorithm and claimed that the CSO algorithm provides better results. Wang et al. [31] proposed a new algorithm based on CSO and called it NCSO. In their work, a new mechanism was introduced to make the MR parameter dynamic. Further, the Cauchy mutation operator was applied to enhance the global search of CSO algorithm. The performance of the NCSO algorithm was tested on twelve benchmark functions, and it was found that the NCSO is an effective and efficient algorithm. Kotekar and Kamath applied the CSO algorithm for clustering the web documents, and their results showed that the CSO algorithm enhances the web service process [32]. Yusiong applied the CSO algorithm to optimize the parameters of the artificial neural network (ANN). It was reported that the CSO is an effective tool for optimizing the parameters of ANN. Sharafi et al. [33] developed a binary version of CSO algorithm and tested it on several benchmark optimization functions. The results showed that the proposed algorithm obtains better quality results in comparison to the GA and Binary PSO algorithms. Pappula and Ghosh applied CSO algorithm for the synthesis of linear antenna arrays. The simulation results showed that the CSO algorithm gives better results than the PSO and ACO algorithms [34]. To improve the accuracy rate, Orouskhani et al. [35] developed a new adaptive dynamic CSO algorithm, which was tested on six benchmark test functions. It was revealed that it takes less time to converge. In literature, the CSO algorithm has also been applied for solving clustering problems [36–38].

3 Cat swarm optimization

Chu and Tasi developed CSO algorithm based on the two key characteristics of cats [19], i.e., hunting and resting skills. According to the hunting skill, a cat has strong curiosity toward moving objects. In the resting skill, a cat spends most of its time in the resting position, even though it remains alert and slowly moves to different positions. But, if a target is identified, then cat quickly captures the target spending a lot of energy. So, on the behalf of these two key characteristics of cats, a mathematical model was formed to solve complex optimization problems and was named cat swarm optimization. In this model, two modes, i.e., seeking and tracing modes, are described to measure the behavior of cats. The working of these modes is explained in Sections 3.1 and 3.2.

3.1 Seeking mode

The seeking mode describes the resting skill of cats. In seeking mode, a cat moves to different positions in the search space, but remains alert. It can be interpreted as local search for the solutions. The following notations are used in this mode.

- Seeking Memory Pool (SMP): This parameter describes the number of copies of a cat to be replicated.
- Seeking Range of selected Dimension (SRD): It denotes the difference between new and old dimensions of cat selected for mutation.
- Counts of Dimension to Change (CDC): It represents the number of dimensions a cat position undergone for mutation.

The steps of seeking mode of CSO algorithm are given as follows.

1. Define the number of copies (T) of i^{th} cat.
2. According to CDC parameter, do the following
   i. Randomly add or subtract SRD values from current position of cats.
   ii. Replace the old values for all copies.
3. Compute the fitness for all copies
4. Choose the best candidate solution and deploy at the position of i^{th} cat.

3.2 Tracing mode

This mode reflects the hunting skill of cats. When a cat hunts the prey, the position and velocity of cat are updated. So, a large difference occurs between new and old positions of cats. The position (X_j^d) and velocity (V_j^d) of the j^{th} cat in the D-dimensional space can be defined as $X_j^d = \left\{X_j^1, X_j^2, \ldots\ldots X_j^D\right\}$; $V_j^d = \left\{V_j^1, V_j^2, \ldots\ldots V_j^D\right\}$; where $d = 1, 2, \ldots D$.

The best position of the cat is represented using $X^d_{best} = \{X^1_{best}, X^2_{best}, \ldots\ldots X^D_{best}\}$.

The velocity and position of the j^{th} cat are computed using (1) and (2).

$$V^d_{j\,new} = w * V^d_j + c * r * \left(X^d_{jbest} - X^d_j\right) \tag{1}$$

where, $V^d_{j\,new}$ represents the updated velocity of j^{th} cat in the d^{th} dimension, w denotes a weight factor in the range of 0 and 1, V^d_j represents the old velocity of the j^{th} cat, c is a user defined constant, r denotes a random number in the range of 0 and 1, X^d_{jbest} represents the best position achieved by j^{th} cat in d^{th} dimension, and X^d_j denotes the current position of the j^{th} cat in d^{th} dimension where $d = 1, 2, \ldots D$.

$$X^d_{j\,new} = X^d_j + V^d_j \tag{2}$$

where, $X^d_{j\,new}$ denotes the updated position of the j^{th} cat in d^{th} dimension, X^d_j denotes the current position of the j^{th} cat in d^{th} dimension and V^d_j represents the velocity of the j^{th} cat.

Mixture Ratio (MR) is used to combine the seeking and tracing modes of the CSO algorithm. MR is designed to determine the number of cats in seeking and tracing modes. The steps of the CSO algorithm are as follows.

1. Initialize the population of cats.
2. Define the user-defined parameters and numbers of cats in seeking mode and tracing mode according to MR parameter value.
3. Compute the fitness function for each cat and memorize the best position.
4. According to flag:
   - If cat is in seeking mode, apply the seeking mode process.
   - Otherwise, apply tracing mode process.
5. Again set the number of cats in tracing and seeking modes according to the MR parameter.
6. Repeat steps 3–5 until the termination condition is satisfied.

4 Proposed ICSO algorithm

According to Orouskhani et al., the CSO algorithm suffers from premature convergence due to its weak diversity and introduced the concept of inertia weight to overcome this issue [23]. Moreover, it is also stated that in CSO algorithm, positions of cats are updated by using the current positions and velocities of cats. Sometime, the algorithm fails to explore promising solutions due to the lack of information regarding global best position of cat. Hence, to deal with these issues, the following amendments are proposed in the CSO algorithm.

- To explore more promising solution and enhance the convergence rate, the global best position of cat is used to guide the positions of cats in tracing mode. Hence, a new modified search equation is proposed for tracing mode of CSO algorithm which includes the global best position of cat.

$$X^{d+1}_{j\,new} = (1 - \beta) * X^d_j + \beta * P_g + V^d_j \tag{3}$$

- The CSO algorithm uses a velocity vector and previous position of cat to update the position of a cat in tracing mode. The updated position of a cat is only influenced by velocity vector. Hence, to improve the diversity of CSO algorithm, especially in tracing mode, a new velocity updated equation is proposed, which is inspired from [39].

$$V^{d+1}_{j\,new} = V^d_j + \beta\left(P_g - X^d_j\right) + \alpha * \varepsilon \tag{4}$$

- where, ε is a random vector uniformly distributed in the range [0, 1]; α and β are acceleration parameters used to direct the position of a cat toward local and global best positions and P_g presents the global best position of a cat
- To make the balance between the exploration and exploitation processes, both of acceleration parameters α and β act as control parameters. The α parameter acts as decreasing function, whereas β parameter serves as an increasing function. In this work, the values of both the parameters are adaptive and computed using the following equations.

$$\alpha(t) = \alpha_{max} - \left\{\frac{\alpha_{max} - \alpha_{min}}{t_{max}}\right\} * t \tag{5}$$

In (5), α_{max} and α_{min} present the upper and lower limits, t_{max} denotes the maximum number of iterations and t denotes the current iteration number. Hence, $\alpha(t)$ is a step function whose value ranges between upper and lower limits. The larger value of α supports exploration whereas small values support exploitation. The aim of $\alpha(t)$ parameter is to control the exploration process of cats in search space.

$$\beta(t) = \beta_{min} + (\beta_{max} - \beta_{min}) \sin\left\{\frac{\pi t}{t_{max}}\right\} \tag{6}$$

In (6), β_{min} and β_{max} denote the minimum and maximum values of first and last iterations respectively

t_{max} presents the maximum number of iterations and t denotes the current iteration number. The reason behind the incorporation of β (t) parameter is to influence the global exploration ability of the proposed algorithm. A large value of parameter strengthens the global best position of cat and also tends to the solution refinement.

4.1 Local search method

In this subsection, a local search method is summarized to improve the quality of solution. The proposed method is applied to the tracing phase of ICSO algorithm. The aim of this method is to guide the search direction and achieve the optimum solution. The need for local search method can be summarized as follows:

- To guide the search direction and obtain the optimum solution in search space
- To overcome the local optima problem through neighborhoods information

This search mechanism is applied to the current global best solution (X_{gbest}), and the neighborhood of best solution can be defined using (7).

$$\text{Neighbor of } \left[X_{gb}\right] = \left[X_{gbest} - r, X_{gbest} + r\right]^n \tag{7}$$

where, "r" describes the boundary of neighborhood, X_{gb} presents the current best solution and can be denoted as X_{gb} (0), and n presents the number of population. The concept of neighborhood can be explained with the help of Fig. 1 as depicted below.

In ICSO, the candidate solution is defined as $\left\{X_j^1(k), X_j^2(k), \ldots\ldots X_j^L(k), \ldots\ldots\ldots, X_j^N(k)\right\}$, where $k = 1, 2, \ldots\ldots, P$. Here, p denotes the number of iterations in the local search method. The local search method can visit N number of population in every iteration such as $X_j^L(k)$ $(L = 1, 2, \ldots\ldots, N)$. It is assumed that $X_{gbest}^L(k) = \left\{x_j^1(k), \ldots x_j^L(k), \ldots, x_j^N(k)\right\}$ and is determined using (8).

$$X_{gb}^L(k) = \begin{cases} X_{gb}^L(k-1) & d \neq L \\ X_{gb}^L(k-1) + r * cx^d & d = L \end{cases} \tag{8}$$

where, $X_{gb}^L(k)$ denotes the global best position of cat in kth iteration where $k = 1, 2, \ldots\ldots, P$ and $L = 1, 2, \ldots, N$; cx^d can be computed using $\left[1 - cx^d(k-1)\right]$ and $d = 1, 2, \ldots\ldots, N$. In each iteration, the best agent during the search can be selected using (9).

$$X_{gb}(k) = MIN\left\{x_{gb}^1(k), \ldots x_{gb}^L(k), \ldots, x_{gb}^n(k)\right\} \tag{9}$$

The above steps are repeated until a better solution is obtained. The procedure of the local search method is illustrated in Algorithm 1.

Algorithm 1 Local search method

For L = 1 : n

Initialize the variables $x^L(k) = rand(0, 1)$

End For

While(termination criteria not met), do the following

For L=1 : n

Determine candidate solution $\left(X_{gbest}^L(k)\right)$*using* (8).

End For

Pick local best candidate solution using (9).

End While

4.2 Steps of ICSO algorithm

In this subsection, the main steps of the proposed CSO algorithm are summarized.

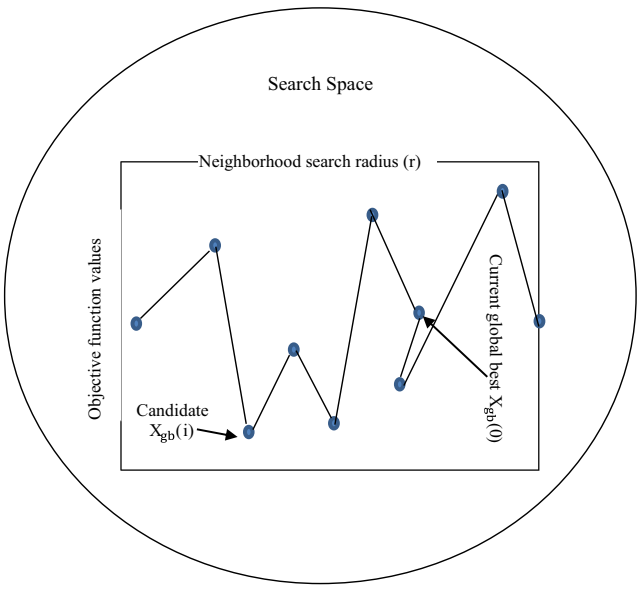

Fig. 1 Illustration of the local search method incorporated into ICSO algorithm

Steps of ICSO algorithm

Step 1: *Initialize the different parameters of the proposed algorithm like number of cats (N), SMP, SRD, neighborhood structure, β, α and C and randomly placed N number of cats in random space search.*
Step 2: *Initialize position and velocity of each cat into the D-dimensional search space.*
Step 3: *Compute the fitness function of cats and store the best position of cat into memory.*
Step 4: *While($i < maximum_{iteration}$)*
Step 5: *According to the value of Flag, randomly distribute cats into tracing and seeking modes*
Step 6: *If (Flag==1); Cat in seeking mode*
Step 7: *For each cat, apply seeking mode process*
Step 7.1: Make j copies of each cat.
Step 7.2: Compute shifting bit value for each cat using SRD.
Step 7.3: Add or subtract each cat to shifting value.
Step 7.4: Compute the fitness function for each new position of cats.
Step 7.5: Compare the value of fitness function and keep the best position of cat into memory.
Step 7.6: End for
Step 8: *Else, Cat in tracing mode*
Step 9: *For each cat, apply tracing mode process*
Step 9.1: Update the velocity of each cat using (3).
Step 9.2: Update the position of each cat using (4).
Step 9.3: Compute the fitness function for newly generated position of cat.
Step 9.4: Compare the fitness function value and keep the best position of cat in memory.
Step 9.5: End for
Step 10: *Update the position of cats and also determine the best position of cat.*
Step 11: *IF ($rand(0, 1) \leq Fit_i$) then*
Step 12: *Apply Local Search Method (Algorithm 1).*
Step 13: *Update the position of cats and global best position.*
Step 14: *End if*
Step 15: *$i = i + +$*
Step 16: *End while*
Step 17: *Obtain the final solution.*

5 Experimental settings

This section describes the experimental results of the proposed CSO algorithm. In this work, two types of problems are considered to evaluate the efficiency of ICSO algorithm; first is standard benchmark test functions and another is clustering problems. In order to test the performance of ICSO algorithm, twelve bench mark test functions are considered. These are the well-known test functions reported in the literature and widely used to assess the performance of algorithms. In this work, both the separable and inseparable unimodal and multimodal functions are taken to examine the performance of ICSO algorithm. The detailed description of these test functions is given in Table 1. The proposed algorithm is implemented in Matlab 2010a environment using windows 7 operating system, Intel corei5 processor, 3.4 GHz and 8 GB RAM. The results are taken on an average of 30 independent runs for each function. Further, the results of the proposed algorithm are compared with several other meta-heuristic algorithms.

5.1 Results and discussion

To start the experiment, it is necessary to set the different parameter values. The parameters of ICSO are as follows: SMP = 10, MR = 0.5, C = 2, $\beta \in 0.1 \sim 0.7$, $\alpha \in 0.1 \sim 0.5$, the size of population is 100 and ε in the range [01]. These parameters are kept constant throughout the execution of the program. The results are taken on average of 30 independent runs. The average and standard deviation parameters are considered as performance measures to evaluate the performances of algorithms. The average parameter indicates the efficiency of algorithms, whereas robustness is computed using the standard deviation parameter. The performance of ICSO is also compared with several popular meta-heuristic algorithms like PSO, GA, ABC, BBO, BAFA, FPA, HS, DE, SFLP, TLBO, DE, and CSO.

The performance of the proposed algorithm is also compared with several variants of GA, DE, HS, SFLP, and TLBO algorithms. Apart from these, the performance of the proposed ICSO algorithm is also compared with some CSO variants like CSO, BCSO, AICSO, and EPCSO. The parameter configurations of above-mentioned algorithms are taken as reported in the corresponding literature [40–47]. Table 2 demonstrates the results of CSO, BCSO, AICSO, EPCSO, and ICSO algorithms with twelve benchmark functions. The average and standard deviation parameters are taken to evaluate the performances of these algorithms. It is observed that ICSO algorithm provides better performance with all of benchmark functions in comparison to other CSO variants. For F_1 and F_6 functions, the proposed algorithm obtains optimum minimum value i.e. 0. For the rest of the functions, it is also seen that ICSO algorithm achieves a minimum value lower than that of the CSO variants.

Table 3 illustrates the results of the proposed ICSO and several popular meta-heuristic algorithms such as PSO,

Table 1 List of test functions used for experimentation

No.	Function name	Definition	Parameter
F_1	Sphere	$F_1(x) = \sum_{i=0}^{D} x_i^2$	[− 100, 100]
F_2	Rosenbrock	$F_2(x) = \sum_{i=1}^{D} 100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2$	[− 30, 30]
F_3	Rastrigin	$F_3(x) = \sum_{i=1}^{D} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$	[− 5.12, 5.12]
F_4	Griewank	$F_4(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	[− 600, 600]
F_5	Ackley	$F_5(x) = 20 + e - 20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right)$	[− 32, 32]
F_6	Step	$F_6(x) = \sum_{i=0}^{D} (x_i + 0.5)^2$	[− 100, 100]
F_7	Powell	$F_{10}(x) = \sum_{i=1}^{D/4} (x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - 10x_{4i})^2 + (x_{4i-2} + 10x_{4i-1})^4 + 10(x_{4i-3} + 10x_{4i})^4$	[− 4, 5]
F_8	Schwefel	$F_8(x) = 418.9828D - \sum_{i=1}^{D} \left(x_i \sin\left(\sqrt{\lvert x_i\rvert}\right)\right)$	[− 500, 500]
F_9	Schaffer	$F_9(x) = 0.5 + \frac{\sin^2\left(x_1^2 - x_2^2\right) - 0.5}{\left[1 + 0.001\left(x_1^2 - x_2^2\right)\right]^2}$	[− 100, 100]
F_{10}	Zakharov's	$F_{10}(x) = \sum_{i=0}^{D} (x_i)^2 + \left(\frac{1}{2}\sum_{i=0}^{D} i x_i\right)^2 + \left(\frac{1}{2}\sum_{i=0}^{D} i x_i\right)^4$	[− 5, 10]
F_{11}	Michalewicz	$F_{11}(x) = \sum_{i=1}^{D} \mathrm{Sin} x_1 \left(\sin\left(i x_i^2 / \pi\right)\right)^{20}$	$[0, \pi]$
F_{12}	Quartic	$F_{12}(x) = \sum_{i=1}^{D} i x_i^4 + \mathrm{rand}(0, 1)$	[− 1.28, 1.28]

Table 2 Comparison of results of CSO and proposed CSO algorithms for benchmark functions (using D=30)

Function	Parameters	CSO	BCSO	AICSO	EPCSO	ICSO
F_1	Average	2.00E-04	1.05E-05	7.01E-05	4.42E-06	0.00E+00
	SD	2.00E-04	1.48E-04	1.31E-04	3.32E-05	0.00E+00
F_2	Average	2.96E+01	2.68E+01	2.54E+01	2.15E+01	1.65E+01
	SD	1.84E+01	1.49E+01	1.33E+01	2.01E+01	1.35E+01
F_3	Average	1.83E-01	1.73E-01	6.63E-02	1.64E-02	5.36E-03
	SD	1.10E-01	1.12E-01	5.45E-02	1.19E-01	4.09E-02
F_4	Average	1.19E-01	9.77E-02	6.91E-02	3.76E-02	3.93E-04
	SD	1.38E-02	6.79E-02	3.17E-02	2.47E-02	2.66E-05
F_5	Average	2.46E-01	2.38E-01	1.98E-01	3.78E-03	6.48E-05
	SD	1.56E-02	2.01E-01	1.47E-02	8.72E-02	4.28E-04
F_6	Average	4.00E-04	5.98E-05	4.91E-05	8.46E-06	0.00E+00
	SD	2.00E-04	9.74E-06	3.86E-05	7.24E-06	0.00E+00
F_7	Average	2.29E+02	2.20E+02	2.15E+02	1.05E+02	7.84E+01
	SD	8.92E+01	6.55E+01	2.15E+01	1.68E+01	4.16E+00
F_8	Average	7.96E-02	6.13E-02	9.63E-03	5.35E-03	5.49E-05
	SD	2.46E-02	5.80E-02	3.57E-03	2.78E-03	1.94E-06
F_9	Average	2.09E+02	1.95E+02	1.84E+02	1.65E+02	7.30E+01
	SD	1.96E+02	6.90E+01	9.63E+01	5.60E+01	1.63E+01
F_{10}	Average	9.22E-13	3.37E-13	5.36E-13	8.56E-14	2.68E-16
	SD	5.32E-14	1.42E-14	1.08E-13	5.21E-14	4.24E-17
F_{11}	Average	− 5.76E+00	4.56E+00	4.67E+00	7.30E-01	− 6.33E-02
	SD	4.16E-01	9.12E-01	3.35E+00	4.33E-01	4.96E−3
F_{12}	Average	3.46E-05	3.18E-05	1.43E-05	3.09E-06	7.84E-08
	SD	2.63E-05	1.77E-06	1.32E-05	2.99E-06	3.41E-07

Table 3 Comparison of ICSO and other meta-heuristic algorithms for standard benchmark functions (using D=30)

Function No.	Parameters	PSO	ABC	BA	FPA	FA	ICSO
F_1	Average	0.0003	5.92E-04	1	0.4283	3.79E-02	0
	SD	0.0015	5.33E-05	1	0.0826	5.35E-02	0
F_2	Average	25.8537	2.35E+01	38.9874	36.9146	1.72E+02	16.4721
	SD	16.2698	1.36E+01	15.6428	17.578	1.30E+02	13.4716
F_3	Average	0.3582	3.29E-01	0.4266	0.8916	2.29E+01	0.0536
	SD	0.6975	4.25E-02	1	0.7538	7.14E+00	0.0409
F_4	Average	0.1045	1.51E-02	0.8205	1	1.05E-01	0.0393
	SD	0.0462	1.24E-02	0.0814	0.0209	5.45E-02	0.0266
F_5	Average	0.521	9.63E-02	1	0.3168	2.05E+00	0.0648
	SD	0.0406	7.86E-02	1	0.0736	3.57E-01	0.0428
F_6	Average	0.0004	1.46E-02	1	0.2764	3.11E-01	0
	SD	0.0026	3.53E-03	1	0.1967	1.18E-01	0
F_7	Average	385.901	1.38E+02	113.897	86.1969	2.49E+03	78.3548
	SD	161.14	7.64E+01	86.0515	80.7233	2.36E+02	41.562
F_8	Average	0.5126	7.44E-01	0.08245	0.07916	6.33E-02	0.005493
	SD	0.0034	5.47E-01	0.0183	0.03816	1.82E-02	0.1938
F_9	Average	9.82E+03	2.43E+02	2.76E+02	3.53E+02	2.86E+02	1.43E+02
	SD	6. 412E+02	3.61E+01	1.12E+02	1.26E+03	4.50E+01	3.25E+01
F_{10}	Average	6.14E-10	8.54E-14	9.83E-14	5.22E-14	5.34E-11	2.68E-16
	SD	1.35E-11	6.84E-15	4.86E-14	4.80E-14	3.96E-12	4.24E-17
F_{11}	Average	− 2. 96847	− 1.44E+01	− 4.538246	− 3.861285	− 9.63E+00	− 1.8123
	SD	0.65249	3.75E-01	0.71593	0.83529	2.42E-02	4.96E−2
F_{12}	Average	0.004357	5.34E-04	0.0041326	0.0038632	8.55E-04	7.84E-06
	SD	0.000571	2.36E-04	0.007865	0.001547	2.13E-04	3.41E-07

ABC, BA, FPA, and FA. It is seen that the proposed algorithm provides better results for most of benchmark test functions using average and SD parameters. It is stated that the ICSO have good exploration capability to search the optimum solutions. Tables 4 and 5 depict the results of ICSO algorithm and some recent meta-heuristic algorithms like BBO, BBBO, HS, MHS, HSDE, SFLP, MSFLA, TLBO, ITLBO, DE, HSDE, GA, QGA-MPC, and GA-MPC. It is observed that the performance of ICSO algorithm is better than other algorithms being compared with most of benchmark functions. It is stated that the proposed algorithm obtains highly precise results due to enhanced solution search equations and also explore solution search space effectively due to local search method.

Figure 2a–f shows the convergence behavior of CSO and ICSO algorithms for six benchmark functions, i.e., F_1–F_6. It is seen that the proposed ICSO algorithm provides better results in terms of cost and number of iterations. Further, it is observed that the ICSO algorithm converges faster than the CSO algorithm. It is also seen that the local search method improves the efficiency of ICSO algorithm. The problem of trapping in local optima in last iterations is also solved.

5.2 Statistical test

In order to illustrate the efficacy of the proposed ICSO algorithm, some statistical tests are also applied to validate the performance of proposed algorithm. In this work, Friedman test is applied for statistical analysis. The Friedman test is a nonparametric statistical test. In Friedman test, two hypotheses are created, i.e., null hypothesis (H_0) and hypothesis (H_1). The null hypothesis (H_0) stands for no significant difference occurs between the performances of proposed algorithm and other algorithms. In contrast, hypothesis (H_1) claimed that a significant difference occurs between the performances of algorithms [12]. Another important parameter of Friedman test is p-value, which is a probability value that shows whether a statistical test is significant or not; the smaller the p-value, the stronger is the evidence against H_0. Table 6 presents the results of Friedman test and average ranking of algorithms. It is seen that the proposed algorithm obtains minimum rank, i.e., 1 among all other algorithms, whereas CSO algorithm has the worst rank, i.e., 5. The degree of freedom is 4. The p-value obtained from Friedman test is 1.354E-6, which strongly rejects the null hypothesis (H_0). It can be seen that

Table 4 Comparison of ICSO and other meta-heuristic algorithms for standard benchmark functions (using D=30)

Function	Parameter	BBO	BBBO	HS	MHS	HSDE	SFLP	MSFLA	ICSO
F_1	Average	2.40E-03	1.97E-4	5.22E-4	1.63E-4	6.08E-5	1.09E-3	5.79E-5	0
	SD	4.56E-04	2.56E-5	3.2945E-5	2.18E-5	5.33E-5	2.90E-21	3.88E-6	0
F_2	Average	5.79E+00	1.79E+01	1.90E+02	2.73E+01	5.33E+02	3.69E+01	2.73E-02	16.4721
	SD	1.27E+00	5.63E+00	5.16E+01	1.33E+01	3.90E+02	2.23E+01	1.94E-02	13.4716
F_3	Average	7.08E-02	3.07E-03	1.69E+01	8.01E-01	0.00E+00	2.44E+01	2.55E+01	0.0536
	SD	5.56E-02	6.20E-04	2.66E+00	8.30E-01	0.00E+00	5.36E+00	1.03E+01	0.0409
F_4	Average	1.89E-01	1.08E-01	1.60E-01	1.43E-01	1.22E-02	3.52E-02	1.64E-02	3.93E-02
	SD	3.36E-02	2.68E-02	5.34E-02	2.93E-02	8.30E-03	3.79E-03	1.99E-02	2.66E-03
F_5	Average	1.01E-01	4.04E-02	1.09E+00	7.71E-14	4.70E+01	9.76E-03	6.67E+01	6.48E-02
	SD	5.13E-02	4.24E-03	1.36E-01	2.37E-14	1.16E+01	5.21E-02	1.15E+01	0.0428
F_6	Average	1.71E+00	3.67E-01	4.37E+00	0.00E+00	1.59E-01	2.36E-06	5.70E-01	0
	SD	3.71E-01	1.66E-01	9.83E-01	0.00E+00	7.97E-01	3.29E-06	1.88E-01	0
F_7	Average	1.38E+02	8.71E+01	4.59E+02	1.17E+02	9.47E+01	1.65E+02	8.95E+01	7.84E+01
	SD	1.14E+02	1.16E+01	5.86E+01	3.40E+01	3.82E+01	4.38E+01	2.95E+01	4.16E+00
F_8	Average	7.50E-01	6.61E-01	1.53E-01	3.63E-02	8.96E-02	9.76E-01	3.67E-03	5.49E-03
	SD	3.86E-02	7.09E-02	3.10E-02	4.80E-03	2.14E-02	5.03E-01	3.18E-03	1.94E-03
F_9	Average	2.46E+02	1.57E+02	2.83E+02	1.84E+02	1.65E+02	2.62E+02	1.77E+02	1.43E+02
	SD	6.47E+01	3.01E+01	1.06E+02	7.42E+01	6.53E+01	1.12E+02	5.97E+01	3.25E+01
F_{10}	Average	1.30E-03	2.34E-10	1.03E-11	4.93E-13	1.53E-15	3.29E-12	2.28E-15	2.68E-16
	SD	1.85E-03	4.28E-11	1.31E-12	6.11E-14	1.06E-15	1.30E-12	5.16E-15	4.24E-17
F_{11}	Average	− 3.92E+00	− 2.16E+00	− 4.32E+00	− 1.72E+00	− 2.42E+00	− 2.95E+00	− 2.43E+00	− 1.8123
	SD	1.31E+00	6.11E-01	1.06E+00	5.16E-01	1.23E+00	4.93E-01	9.28E-01	4.96E−2
F_{12}	Average	2.00E-05	4.94E-06	8.44E-04	2.18E-44	2.27E-03	3.79E-03	2.84E-03	7.84E-06
	SD	3.14E-06	1.03E-06	2.21E-05	6.08E-44	1.06E-03	7.45E-04	1.53E-04	3.41E-07

Table 5 Comparison of ICSO and other meta-heuristic algorithms for standard benchmark functions (using D=30)

Function	Parameter	TLBO	ITLBO	DE	HSDE	GA	QGA-MPC	GA-MPC	ICSO
F_1	Average	0	0	1.35E+02	6.08E–4	0.8378	1.43E+01	5.60E+06	0
	SD	0	0	7.03E+00	8.33E-5	0.5138	2.25E+00	5.64E+05	0
F_2	Average	26.6567	22.7934	5.87E+04	5.33E+02	45.2843	1.54E+04	1.59E+09	16.4721
	SD	19.456	14.823	8.69E+03	3.90E+02	21.6284	1.36E+04	3.12E+08	13.4716
F_3	Average	0.2178	0.1632	5.17E+02	0.00E+00	1	5.19E+02	1.55E+04	0.0536
	SD	0.06325	0	1.32E+01	0.00E+00	0.6921	3.52E+01	1.55E+03	0.0409
F_4	Average	0	0	1.03E+00	1.22E-02	0.8613	9.56E-01	1.39E+03	0.0393
	SD	0	0	8.17E-02	8.30E-03	0.0648	8.53E-02	1.08E+02	0.0266
F_5	Average	3.55E−01	1.42E−01	3.02E+02	4.70E+01	0.7934	2.12E+01	2.09E+01	0.0648
	SD	0.0836	0.05618	5.20E+00	1.16E+01	0.3216	3.32E-02	3.32E-02	0.0428
F_6	Average	2.74E−02	1.17E−4	1.15E-01	1.59E-01	0.7872	9.08E-03	1.47E-02	0
	SD	5.36E−03	4.68E−5	1.66E-01	7.97E-01	0.5643	3.77E-02	1.58E-02	0
F_7	Average	124.1484	83.7532	1.91E+04	9.47E+01	830.075	1.65E+04	2.83E+03	78.3548
	SD	2.60E+02	1.35E+01	3.69E+03	3.82E+01	393.68	5.09E+02	5.12E+02	41.562
F_8	Average	0.0066	0.0053	1.09E-01	8.96E-02	0.5397	7.83E-02	9.63E-01	0.005493
	SD	4.50E-03	3.20E-03	9.85E-02	2.14E-02	0.0073	2.88E-02	5.33E-01	0.1938
F_9	Average	2.17E+02	1.59E+02	1.29E+03	1.65E+02	4.62E+04	1.74E+02	1.32E+00	1.43E+02
	SD	1.26E+02	6. 832E+01	4.27E+02	6.53E+01	1.59E+03	2.14E+01	5.37E+01	3.25E+01
F_{10}	Average	4.64E-14	3.46E-16	2.60E-10	1.53E-15	5.50E-10	3.27E-15	5.13E-14	2.68E-16
	SD	2.34E-14	1.89E-16	1.43E-11	1.06E-15	7.62E-10	2.33E-16	2.44E-14	4.24E-17
F_{11}	Average	− 4.352678	− 3.56815	− 1.80E+01	− 2.42E+00	− 5.86435	− 3.60E+00	− 5.86E+00	− 1.8123
	SD	1.44E-02	1.53E−2	4.20E-01	1.23E+00	0.91578	6.72E-01	1.05E+00	4.96E−2
F_{12}	Average	3.25E-04	2.19E-05	6.64E-03	2.27E-03	0.235416	4.56E-05	5.34E-05	7.84E-06
	SD	1.59E-04	1.22E-05	4.72E-03	1.06E-03	0.067134	2.90E-05	3.87E-05	3.41E-07

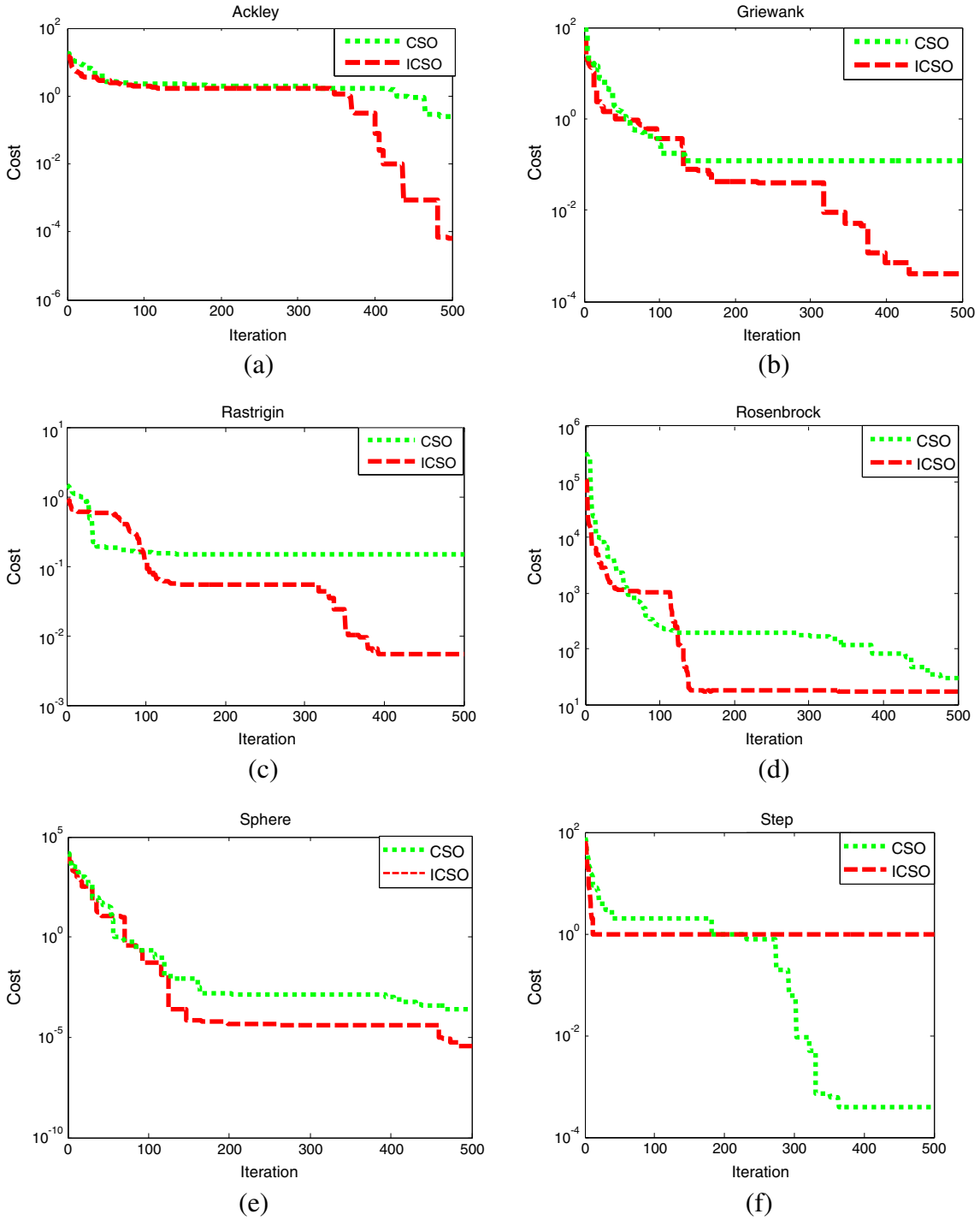

Fig. 2 **a–f** shows the convergence of CSO and ICSO algorithms for functions $F_1 - F_6$

a significant difference occurs between the performances of algorithms. Multiple comparisons (N × N) of all algorithms are also carried out for benchmark test functions. Table 7 presents the results of (N × N) comparisons with adjusted p-values and test statistics. It is noticed that the proposed ICSO algorithm obtains minimum p-values than other algorithms. Overall, it is concluded that the proposed algorithm is an effective and efficient algorithm for solving the complex benchmark test functions than other CSO variants.

Further, Wilcoxon signed-ranks test is applied on the two best-performing algorithms. The two best-performing

Table 6 Friedman test statistics for all algorithm using benchmark test functions

Algorithm	Functions												Overall rank	Mean of rank
	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12		
CSO	5	5	5	5	5	5	5	5	5	5	5	5	60	5
BCSO	3	4	4	4	4	4	4	4	4	3	3	4	45	3.75
AICSO	4	3	3	3	3	3	3	3	3	4	4	3	39	3.25
EPCSO	2	2	2	2	2	2	2	2	2	2	2	2	24	2
Proposed ICSO	1	1	1	1	1	1	1	1	1	1	1	1	12	1

Number of observation: 60 Number of problems: 12 Number of algorithms: 5
Sum of squares of ranks : 7462 Correction factor: 540 Friedman test statistic: 32.73
Degree of freedom: 4 p-value: 1.354E-6 Critical value: 9.4877

Friedman Test Hypothesis: (i) H_0: Algorithms are equally effective. (ii) H_1: There is a significant difference in the performances of algorithms. (significance level = 0.05)

algorithms are EPCSO and proposed ICSO. Wilcoxon signed-ranks test is a non-parametric test applied in hypothetical testing involving two samples [13]. It is a pairwise test that can be used to detect significant differences between the performance of EPCSO and ICSO algorithms. This test is closely related to t-test. The results of this test are mentioned in Tables 8-9. Table 8 provides the details of rank and signed rank using Wilcoxon test, whereas Table 9 shows the statistics of test. The p-value obtained for this test is 0.016113, and the median values for EPCSO and ICSO algorithms are 0.0045653and 5.985E-05, respectively. The statistics value (z) for Wilcoxon test is − 2.35. From the statistical results, it is stated that the hypothesis strongly rejects the null hypothesis at the significance level of 0.05 and a significant difference occurs between the median values of EPCSO and ICSO algorithms.

6 Application of ICSO in data clustering

Data clustering is a powerful data analysis technique that can be used to group the data objects in the form of clusters using distance based similarity function [6]. The similarity function is used to compute the distance between data objects and cluster centers. The data clustering can be formulated as follows. Consider $S = \{s_1, s_2, s_3 \ldots\ldots s_n\}$ is a set of n objects and $T_{n\times m}$ is a data matrix consisting of n rows and p columns, where n corresponds to data objects and m represents the attributes or dimensions of a given dataset. In the given dataset, each i^{th} data object is described by using a real valued m-dimensional vector $x_i \{i = 1, 2, 3, \ldots\ldots n\}$ where each object x_{ij} denotes the j^{th} feature $(j = 1, 2, 3, \ldots .m)$ of the i^{th} object $(i = 1, 2, 3, \ldots .n)$. The goal of the clustering

Table 7 Adjusted p-values and test statistics for multiple comparisons (N × N comparison) among all algorithms

Algorithm	p value	TS	Algorithm	p value	TS	Algorithm	p value	TS
CSO			BCSO			AICSO		
BCSO	2.02E-02	2.4109	CSO	0.020159	2.4109	CSO	0.000143	4.1642
AICSO	0.000143	4.1642	AICSO	0.086507	1.7533	BCSO	8.65E-02	1.7533
EPCSO	4.81E-08	6.5751	EPCSO	0.000143	4.1642	EPCSO	0.020159	2.4109
Proposed ICSO	3.29E-11	8.7667	Proposed ICSO	1.01E-07	6.3559	Proposed ICSO	3.54E-05	4.6025
EPCSO			Proposed ICSO					
CSO	4.81E-08	6.58E+00	CSO	3.29E-11	8.7667			
BCSO	0.000143	4.1642	BCSO	1.01E-07	6.3559			
AICSO	0.020159	2.4109	AICSO	3.54E-05	4.6025			
Proposed ICSO	3.37E-02	2.19E+00	EPCSO	0.033739	2.1917			

Table 8 Ranks of each function using Wilcoxon test for CSO and ICSO algorithms

Function	CSO	ICSO	D	Abs(D)	Rank	Signed-Rank
F_1	4.42E-06	0	− 4.42E-06	4.42E-06	3	− 3
F_2	21.49	16.4721	− 5.02	5.02	10	− 10
F_3	0.0164	0.00536	− 0.01	0.01	7	− 7
F_4	0.0376	0.000393	− 0.04	0.04	8	− 8
F_5	0.0037813	6.48E-05	− 3.72E-03	3.72E-03	5	− 5
F_6	8.46E-06	0	− 8.46E-06	8.46E-06	4	− 4
F_7	104.72	78.4	− 26.32	26.32	11	− 11
F_8	0.0053493	5.49E-05	− 0.01	0.01	6	− 6
F_9	165	73	− 92	92	12	− 12
F_{10}	8.56E-14	2.68E-16	− 8.53E-14	8.53E-14	1	− 1
F_{11}	− 0.73	− 0.06328	0.67	0.67	9	9
F_{12}	3.09E-06	7.84E-08	− 3.01E-06	3.01E-06	2	− 2

problem is to compute a set of optimal partitions $z = \{z_1, z_2, \ldots\ldots z_k\}$ for the given data matrix $T_{n\times m}$ and can satisfy the following conditions.

i. $\forall_{i\neq j} z_i \cap z_j = \emptyset$,
ii. $\bigcup_{i=1}^{k} z_i = x$,
iii. $\forall_i z_i = \emptyset$

It is observed that the Euclidean distance can be used as similarity function to compute the set of optimal partitions. It can be defined as the distance between data objects and the cluster centers. It is computed for every cluster center and assigned to all data objects. The data is clustered on the basis of minimum Euclidean distance. The objective function for clustering problems can be described as follows.

$$\text{minimize } F(X, C) = \sum_{k=1}^{K} \sum_{x\in D_i} \min \|X_i - C_k\|^2 \tag{10}$$

In (10), X_i denotes the i^{th} data object, C_k represents the k^{th} and data objects are assigned to clusters according to the minimum distance. After partitioning the data into different clusters, a fitness function is applied to determine the goodness of the partitions. In this work, sum of square error (SSE) based fitness function is adopted to determine the fitness of each cluster. This fitness function is described in (11).

$$F(C_k) = \sum_{k\in 1}^{K} \frac{SSE(C_k)}{\sum_{k=1}^{K} SSE(C_K)} \tag{11}$$

Further, the proposed algorithm is also applied to solve the clustering problems. The main motive of the ICSO algorithm is to find optimal set of cluster for clustering problems. To evaluate the performance of the proposed algorithm, some benchmark datasets are taken from UCI repository. The results of the ICSO algorithm is also compared with well-known clustering algorithms reported in the literature.

Table 9 Results of Wilcoxon test (paired samples) on CSO and ICSO algorithms

Test Statistics	CSO	ICSO
Mean	24.21	13.98
Median	0.0045653	5.985E-05
Significant value	0.05	
p-value	0.016113	
Sum of signed Rank (W)	60	
E (W)	0.5	
SD (W)	25.495098	
Z	0.019607	
Hypothesis- H_0: Median 1 is equal to Median 2; H_1: Median 1 is not equal to Median 2.		
Hypothesis (H_0) is rejected at the significance level 0.05		

6.1 Steps of ICSO algorithm for clustering

This subsection describes the algorithmic steps of the ICSO algorithm for solving clustering problems The main steps of ICSO algorithm are as follows.

Table 10 Description of datasets used for experiment

Dataset	Cluster (K)	Features	Total Data Items	Data in each cluster
Iris	3	4	150	(50, 50, 50)
Wine	3	13	178	(59, 71, 48)
CMC	3	9	1473	(629,334, 510)
Cancer	2	9	683	(444, 239)
Glass	6	9	214	(70,17, 76, 13, 9, 29)

Algorithm 2 Structure of proposed ICSO algorithm for clustering problems

%% Initialization Steps %%

Step 1: *Load the dataset, initialize the parameters of ICSO algorithm such as population of cats (no. of cluster centers K), SMP, SRD, c_1, r_1, maximum no. of generation*

Step 2: *For each k=1 to K/* K represents the total number of cluster centers */*
For each j=1 to D/ D represents the total number of features in dataset */*
Determine the initial positions of cats, i.e., cluster centers using equation

$$C_k = X_{min,d} + (k-1)\left(X_{max,d} - X_{min,d}\right)/K \tag{12}$$

End For
Determine the velocity (V_K)*of each cat.*
End For

Step 3: *For each k=1 to K/* K represents the total number of cluster centers */*
For each j=1 to D/ D represents the total number of features in dataset */*
Determine the value of objective function using (10);
Assign the data objects (X_i)*to clusters*(C_K)*such that*

$$\left(C_{ik} \text{ and } Z_p\right) = \forall K = \textit{minimize } F(X, C) \tag{13}$$

Compute the fitness function for each cat, i.e., cluster using (11).
End For
End For

Step 4: *While (stopping condition is not met), do the following*

Step 5: *If flag is on; apply seeking mode of ICSO, otherwise tracing mode of ICSO*

%% Seeking mode process of ICSO algorithm %%

Step 6: **/ Replicate "n" number of copies of the present position of cat (cluster center) where "n" is equal to the value of SMP parameter */*
For each cat j=1: K
Replicate the current position of k^{th} *cat to the value of SMP.*
End for

Step 7: *Compute the shifting value for each replicated position of cat using SRD*cluster center (k).*

Step 8: *Randomly add or subtract shifting value with cluster center (k).*

Step 9: *Compute the value of objective function using updated positions of cats.*

Step 10: *Compute the value of fitness function and keep the best position of cat in* X_{Sbest} *and the value of Fitness function* (SSE_S).

Step 11: *Else; cat in tracing mode*

%% Tracing mode process of ICSO algorithm %%

Step 12: *:/* Compute the updated velocity and position for each cat.*/*
For each cat j=1: K
Update the velocity of k^{th} *cat using* (4).
Update the position of k^{th} *cat using the* (3).
End for

Step 13: *Compute the value of objective function using updated positions of cats.*

Step 14: *Compute the value of fitness function and keep the best position of cat in* X_{Tbest} *and the value of Fitness function* (SSE_T).

Step 15: *Update the global best positions of cats by comparing seeking and tracing mode fitness function; If* $(SSE_T \leq SSE_S)$*, then* $X_{gbest} \rightarrow X_{Tbest}$ *and* $SSE_g \rightarrow SSE_T$*; Else* $(SSE_S \leq SSE_T)$*, then,* $X_{gbest} \rightarrow X_{Sbest}$ *and* $SSE_g \rightarrow SSE_S$

Step 16: *If* $\left(rand(0, 1) \leq SSE_g\right)$ *then*

Step 17: *Apply Local Search Method (Algorithm 1).*

Step 18: *Update the position of cats and global best position.*

Step 19: *End If*

Step 20: $i = i++$

Step 21: *End while*

Step 22: *Obtain final optimal solution, i.e., cluster centers*

Table 11 Comparison of performance of the proposed ICSO and other clustering algorithms

Dataset	Parameters	Algorithms					
		K-means	PSO	ACO	CSO	TLBO	ICSO
Iris	Best Case	97.12	96.48	96.89	96.94	96.56	95.78
	Avg. Case	112.44	98.56	98.28	97.86	96.84	97.05
	Worst Case	122.46	99.67	99.34	98.58	98.08	97.63
	Standard Deviation	15.326	0.467	0.426	0.392	0.546	0.213
	F-Measure	0.781	0.78	0.778	0.781	0.782	0.784
Cancer	Best Case	2989.46	2978.68	2983.49	2985.16	2876.28	2943.24
	Avg. Case	3248.25	3116.64	3178.09	3124.15	3084.74	3036.41
	Worst Case	3566.94	3358.43	3292.41	3443.56	3215.83	3253.56
	Standard Deviation	256.58	107.14	93.45	128.46	42.11	63.04
	F-Measure	0.832	0.826	0.829	0.831	0.834	0.835
CMC	Best Case	5828.25	5792.48	5756.42	5712.78	5778.61	5654.11
	Avg. Case	5903.82	5846.63	5831.25	5804.52	5836.25	5761.48
	Worst Case	5974.46	5936.14	5929.36	5921.28	5921.32	5896.25
	Standard Deviation	49.62	48.86	44.34	43.29	38.96	45.16
	F-Measure	0.337	0.333	0.332	0.334	0.331	0.339
Wine	Best Case	16768.18	16483.61	16448.35	16431.76	16578.42	16296.44
	Avg. Case	18061.24	16417.47	16530.53	16395.18	16360.04	16342.21
	Worst Case	18764.49	16594.26	16616.36	16589.54	16917.26	16483.62
	Standard Deviation	796.13	88.27	48.86	62.41	56.14	36.06
	F-Measure	0.519	0.516	0.522	0.521	0.52	0.526
Glass	Best Case	222.43	264.56	273.22	256.53	246.89	249.25
	Avg. Case	246.51	278.71	281.46	264.44	256.44	268.45
	Worst Case	258.38	283.52	286.08	282.27	287.52	281.73
	Standard Deviation	18.32	8.59	6.58	15.43	15.29	10.34
	F-Measure	0.426	0.412	0.402	0.416	0.422	0.427

6.2 Performance evaluation

To investigate the performance of the proposed algorithm, the results of ICSObased clustering algorithm are compared with well-known clustering algorithms such as K-Means, GA, PSO, TLBO, ACO and CSO algorithms [38, 48–51] Five real-life datasets are taken from the UCI database to evaluate the performance of all abovementioned algorithms. Table 10 presents the details of these datasets. The quality of clusters is measured in terms of intra cluster distance (best, average and worst), standard deviation and f-measure parameters. Large values of these measures are required for better clustering. The parameters of the algorithms are the same as reported in the literature. Table 11 shows the comparison between the proposed ICSO algorithm and the other techniques in terms of cluster-quality measures. The results are compared in terms of means and standard deviation over 1 independent runs in each case. For iris, cancer, CMC, wine and glass datasets, it is found that the ICSO algorithm provides better results than the other algorithms being compared Figs. 3 and 4 show the clustering on iris data using ICSO algorithm.

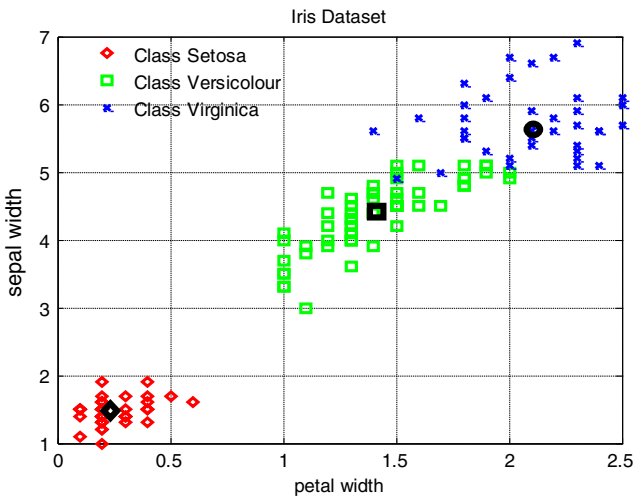

Fig. 3 Clustering of iris dataset using ICSO algorithm (2D View)

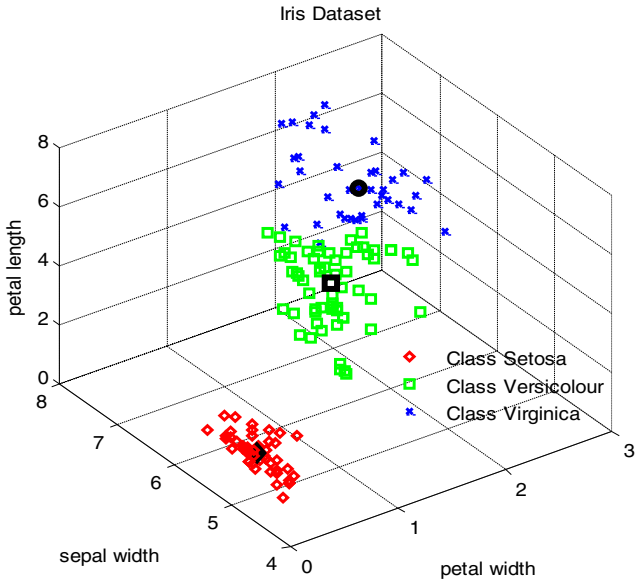

Fig. 4 Clustering of iris dataset using ICSO algorithm (3D View)

7 Conclusion

In this paper, an improved version of CSO algorithm is proposed using improved solution search equations for exploring optimal solution. In addition, two parameters are also introduced for better tradeoff between exploration and exploitation processes. Further, a local search method is also incorporated in the CSO algorithm for avoiding premature convergence problem and handling local optima problem. These improvements in CSO algorithm also improve the precision of solutions. The performance of the proposed algorithm is evaluated on twelve benchmark test functions and real life clustering problems. It is observed that the proposed algorithm gives better results in comparison to other existing algorithms. The simulation results demonstrate the ability and efficacy of the proposed algorithm for solving benchmark test functions. Moreover, the proposed algorithm is also implemented to solve some real life problems. Clustering is one of the reallife problems in which determining the optimum cluster center is a tough task. In clustering problems, the proposed ICSO algorithm is used to determine optimum cluster centers. It is observed that ICSO algorithm provides stateof-the-art results for clustering problems compared to other algorithms. The proposed ICSO algorithm performs better clustering in terms of quality of clusters.

References

1. Stutzle TG (1998) Local search algorithms for combinatorial problems: analysis, improvements, and new applications. PhD Thesis, Technical University of Darmstadt, Darmstadt, Germany
2. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680
3. Holland JH (1992) Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT Press, Cambridge
4. Eberhart R, Kennedy J (1995) A new optimizer using particle swarm theory. In: Proceedings of the sixth international symposium on micro machine and human science, 1995. MHS'95. IEEE, pp 39–43
5. Dorigo M, Birattari M, Stutzle T (2006) Ant colony optimization. IEEE Comput Intell Mag 1(4):28–39
6. Lee KS, Geem ZW (2004) A new structural optimization method based on the harmony search algorithm. Comput Struct 82(9):781–798
7. Karaboga D, Basturk B (2007) Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems. Foundations of fuzzy logic and soft computing, 789–798
8. Yang XS (2009) Firefly algorithms for multimodal optimization. In: International symposium on stochastic algorithms. Springer, Berlin, pp 169–178
9. Kashan AH (2011) An efficient algorithm for constrained global optimization and application to mechanical engineering design: league championship algorithm (LCA). Comput Aided Des 43(12):1769–1792
10. Eskandar H, Sadollah A, Bahreininejad A, Hamdi M (2012) Water cycle algorithm–a novel metaheuristic optimization method for solving constrained engineering optimization problems. Comput Struct 110:151–166
11. Kaveh A, Talatahari S (2010) A novel heuristic optimization method: charged system search. Acta Mech 213(3–4):267–289
12. Kumar Y, Sahoo G (2014) A charged system search approach for data clustering. Progress Artif Intell 2(2–3):53–166
13. Kaveh A, Share AMAM, Moslehi M (2013) Magnetic charged system search: a new meta-heuristic algorithm for optimization. Acta Mech 224(1):85–107
14. Kumar Y, Sahoo G (2015) Hybridization of magnetic charge system search and particle swarm optimization for efficient data clustering using neighborhood search strategy. Soft Comput. https://doi.org/10.1007/s00500-015-1719-0
15. Kumar Y, Gupta S, Sahoo G (2016) A clustering approach based on charged particles. International Journal of Software Engineering and Its Applications 10(3):9–28
16. Rao RV, Savsani VJ, Vakharia DP (2011) Teaching-learning-based optimization: a novel method for constrained mechanical design optimization problems. Comput-Aided Des 43(3):303–315
17. Sahoo AJ, Kumar Y (2014) Modified teacher learning based optimization method for data clustering. In: Advances in signal processing and intelligent recognition systems. Springer International Publishing, pp 429–437
18. Sadollah A, Bahreininejad A, Eskandar H, Hamdi M (2013) Mine blast algorithm: a new population based algorithm for solving constrained engineering optimization problems. Appl Soft Comput 13(5):2592–2612
19. Chu SC, Tsai PW, Pan JS (2006) Cat swarm optimization. In: Pacific Rim international conference on artificial intelligence. Springer, Berlin, pp 854–858
20. Mohapatra P, Chakravarty S, Dash PK (2016) Microarray medical data classification using kernel ridge regression and modified cat swarm optimization based gene selection system. Swarm Evol Comput 28:144–160
21. Tsai PW, Pan JS, Chen SM, Liao BY, Hao SP (2008) Parallel cat swarm optimization. In: 2008 international conference on machine learning and cybernetics, vol 6. IEEE, pp 3328–3333
22. Tsai PW, Pan JS, Chen SM, Liao BY (2012) Enhanced parallel cat swarm optimization based on the Taguchi method. Expert Syst Appl 39(7):6309–6319

23. Orouskhani M, Mansouri M, Teshnehlab M (2011) Average-inertia weighted cat swarm optimization. In: International conference in swarm intelligence. Springer, Berlin, pp 321–328
24. Ram G, Mandal D, Kar R, Ghoshal SP (2015) Circular and concentric circular antenna array synthesis using cat swarm optimization. IETE Tech Rev 32(3):204–217
25. Yang F, Ding M, Zhang X, Hou W, Zhong C (2015) Non-rigid multi-modal medical image registration by combining L-BFGS-B with cat swarm optimization. Inf Sci 316:440–456
26. Lin KC, Huang YH, Hung JC, Lin YT (2015) Feature selection and parameter optimization of support vector machines based on modified cat swarm optimization. Int J Distrib Sens Netw 2015:3
27. Guo L, Meng Z, Sun Y, Wang L (2016) Parameter identification and sensitivity analysis of solar cell models with cat swarm optimization algorithm. Energy Convers Manag 108:520–528
28. Liu D, Hu Y, Fu Q, Imran KM (2016) Optimizing channel cross-section based on cat swarm optimization. Water Sci Technol Water Supply 16(1):219–228
29. Ram G, Mandal D, Kar R, Ghoshal SP (2015) Cat swarm optimization as applied to time-modulated concentric circular antenna array: analysis and comparison with other stochastic optimization methods. IEEE Trans Antennas Propag 63(9):4180–4183
30. Nireekshana T, Rao GK, Raju SS (2016) Available transfer capability enhancement with FACTS using cat swarm optimization. Ain Shams Eng J 7(1):159–167
31. Wang ZH, Chang CC, Li MC (2012) Optimizing least-significant-bit substitution using cat swarm optimization strategy. Inf Sci 192:98–108
32. Kotekar S, Kamath SS (2016) Enhancing service discovery using cat swarm optimization based web service clustering. Perspect. Sci. 8:715–717
33. Yusiong JPT (2012) Optimizing artificial neural networks using cat swarm optimization algorithm. International Journal of Intelligent Systems and Applications 5(1):69
34. Sharafi Y, Khanesar MA, Teshnehlab M (2013) Discrete binary cat swarm optimization algorithm. In: 3rd international conference on computer, control & communication (IC4), 2013. IEEE, pp 1–6
35. Orouskhani M, Orouskhani Y, Mansouri M, Teshnehlab M (2013) A novel cat swarm optimization algorithm for unconstrained optimization problems. International Journal of Information Technology and Computer Science (IJITCS) 5(11):32
36. Kumar Y, Sahoo G (2015) A hybrid data clustering approach based on improved cat swarm optimization and K-harmonic mean algorithm. AI Commun 28(4):751–764
37. Kumar Y, Sahoo G (2016) A hybridise approach for data clustering based on cat swarm optimisation. Int J Inf Commun Technol 9(1):117–141
38. Kumar Y, Sahoo G (2015) An improved cat swarm optimization algorithm for clustering. In: Computational intelligence in data mining, vol 1. Springer, India, pp 187–197
39. Yang XS (2010) Engineering optimization: an introduction with metaheuristic applications. Wiley, Hoboken
40. IKhuat TT, Le MH (2016) A genetic algorithm with multi-parent crossover using quaternion representation for numerical function optimization. Appl Intell 1–17
41. Wang HB, Zhang KP, Tu XY (2015) A mnemonic shuffled frog leaping algorithm with cooperation and mutation. Appl Intell 43(1):32–48
42. Yi J, Gao L, Li X, Gao J (2016) An efficient modified harmony search algorithm with intersect mutation operator and cellular local search for continuous function optimization problems. Appl Intell 44(3):725–753
43. Guo W, Chen M, Wang L, Wu Q (2016) Backtracking biogeography-based optimization for numerical optimization and mechanical design problems. Appl Intell 44(4):894–903
44. Yi W, Gao L, Li X, Zhou Y (2015) A new differential evolution algorithm with a hybrid mutation operator and self-adapting control parameters for global optimization problems. Appl Intell 42(4):642–660
45. Tsai PW, Pan JS, Chen SM, Liao BY (2012) Enhanced parallel cat swarm optimization based on the Taguchi method. Expert Syst Appl 39(7):6309–6319
46. Sharafi Y, Khanesar MA, Teshnehlab M (2013) Discrete binary cat swarm optimization algorithm. In: IEEE 3rd international conference on computer, control & communication, pp 1–6
47. Orouskhani M, Orouskhani Y, Mansouri M, Teshnehlab M (2013) A novel cat swarm optimization algorithm for unconstrained optimization problems. International Journal of Information Technology and Computer Science (IJITCS) 5(11):32
48. MacQueen J (1967) Some methods for classification and analysis of multivariate observations. In: Fifth Berkeley symposium on mathematics. Statistics and probability. University of California Press, pp 281–297
49. Maulik U, Bandyopadhyay S (2000) Genetic algorithm-based clustering technique. Pattern Recogn 33(9):1455–1465
50. Van der Merwe DW, Engelbrecht AP (2003) Data clustering using particle swarm optimization. In: The 2003 congress on evolutionary computation, 2003. CEC'03, vol 1. IEEE, pp 215–220
51. Kumar Y, Sahoo G (2017) A two-step artificial bee colony algorithm for clustering. Neural Comput & Applic 28(3):537–551

Dr. Yugal Kumar is presently working as Assistant Professor (Senior Grade) in Department of Computer Science & Engineering at Jaypee University of Information Technology (JUIT), Waknaghat, Himachal Pradesh, India. He has more than 10 years of teaching and research experience at reputed colleges and universities of India. He has completed his Ph.D. in Computer Science & Engineering from Birla institute of Technology, Mesra, Ranchi. His primary area of research includes meta-heuristic algorithms, data clustering, swarm intelligence, pattern recognition, medical data international journals and conferences of repute. He is serving as editorial review board member of various journals including Soft Computing, Neurocomputing, Computer Methods and Programs in Biomedicine, PLOSE ONE, Journal of Advanced Computational Intelligence and Intelligent Informatics and Journal of Information Processing System.

Dr. Pradeep Kumar Singh is currently working as Assistant Professor (Senior Grade) in Department of Computer Science & Engineering at Jaypee University of Information Technology (JUIT), Waknaghat, H.P. He is having 10 years of vast experience in academics at reputed colleges and universities of India. He has completed his Ph.D. in Computer Science & Engineering from Gautam Buddha University (State Government University), Greater Noida, UP, India. He is Senior Member of ACM Society. Dr. Singh is having life membership of Computer Society of India (CSI) as Senior Member Grade. He has worked as publicity chair of five IEEE International Conferences and associated as TPC member & reviewer of various Conferences & Journals too. He is Associate Editor of International Journal of Information Security and Cybercrime (IJISC) a scientific peer reviewed journal from Romania. He is currently guiding four Ph.D. scholars and having 176 Google scholar citations in his account. He has published 35 research papers in various International Journals and Conferences of repute. He is editorial review board member of various Scopus and SCI Journals including IEEE Transactions on Industrial Informatics. IJISP, IJSSE from IGI Global, USA. He has two sponsored research project from HPSCSTE & HIMCOSTE, HP Govt., INDIA.