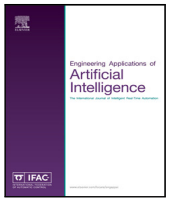
Engineering Applications of Artificial Intelligence 107 (2022) 104506

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



Advanced strategies on update mechanism of Sine Cosine Optimization Algorithm for feature selection in classification problems

Gizem Ataç Kale [a], Uğur Yüzgeç [b,*]

[a] Department of Computer Center, Eskisehir Technical University, Eskisehir, Turkey
[b] Department of Computer Engineering, Bilecik Seyh Edebali University, 11210, Bilecik, Turkey

ARTICLE INFO

Keywords:
Feature selection
Sine Cosine Algorithm
Optimization
Classification
Machine learning

ABSTRACT

Sine Cosine Algorithm (SCA) that is one of the population-based metaheuristic optimization algorithms basically consists of the updating mechanism based on sine and cosine functions. In this algorithm, a few random and adaptive variables are also utilized for more effective motions of the candidate solutions. SCA has some drawbacks like other some metaheuristic algorithms. SCA tends to be stuck into the local regions in the search space and this affects negatively on the computational effort required to find the best solution point in the search space. This paper presents four different improved versions of SCA. The proposed improvements on original SCA are the innovations on the updating mechanism of SCA. To evaluate the performances of Improved Sine Cosine Algorithms (ImpSCAs), well-known numerical optimization problems including CEC 2014 test suite are used. Firstly, different analyses of the proposed ImpSCAs are dealt with such as the convergence analysis, search history analysis, trajectory analysis, average distance analysis, and computational complexity analysis. Secondly, the proposed four versions of ImpSCAs are compared with the original SCA for CEC 2014 benchmark problems with dimension sizes of 10D, 30D and 50D. Finally, original SCA and ImpSCAs are adapted to select optimal feature combination and they are tested for 10 feature selection datasets taken from the UCI machine learning repository. The benchmark results show that the performances of the $ImpSCA_1$, $ImpSCA_2$, and $ImpSCA_4$ are better than that of the original SCA. From the feature selection results, it is observed that three versions of ImpSCAs (except $ImpSCA_3$) outperform the original SCA in 80% of the datasets. Source codes of ImpSCAs are publicly available at https://github.com/uguryuzgec/ImpSCAs.

1. Introduction

Optimization is the process of finding optimal values for the parameters of a system from all the possible values to maximize or minimize its output. Optimization problems are considered as black boxes by stochastic optimization algorithms where outputs are observed only by changing inputs. Since optimization is used in many applications, researchers developed improved optimization techniques. Over the last two decades, the stochastic optimization approaches took a large interest from the researchers (Boussaïd et al., 2013; Bianchi et al., 2009). The metaheuristic algorithms are inspired by natural events such as hunting, social hierarchy, feeding habits, and mating. Many metaheuristics utilize some form of stochastic optimization.

Metaheuristic algorithms are widely used in engineering optimization applications due to their simple concepts, ease of implementation, and no need for derivative operations. Metaheuristic algorithms present optimal solutions for different complex optimization problems, but do not guarantee optimal solutions (Abdollahzadeh et al., 2021a). The reason why such algorithms cannot guarantee the best possible solutions is due to the stochastic search mechanism.

In the past fifty years, the size of the data used in data mining and machine learning has increased rapidly. These extremely high data brought serious difficulties to current learning methods. Having many features leads a learning model to tend to overfit and its learning performance deteriorates. There are several techniques to solve the problem of reducing unrelated and unnecessary variables, known as the curse of dimensionality (Alelyani et al., 2013; Jović et al., 2015). Feature selection is one of the most used techniques and it is a pre-processing step in machine learning. Feature selection removes effectively inappropriate and redundant features. In this way, it can improve the performance of classifiers, reduce the cost of computing and reduce the required storage space (Chandrashekar and Sahin, 2014; Aggarwal, 2014).

Feature selection methods are divided into four categories: filters, wrappers, embedded, and hybrid methods. In the filter methods, the variable ranking techniques are utilized as the principal criteria for variable selection. According to certain criteria, the score of each feature is evaluated (Chandrashekar and Sahin, 2014). The wrapper

* Corresponding author.
E-mail address: ugur.yuzgec@bilecik.edu.tr (U. Yüzgeç).

https://doi.org/10.1016/j.engappai.2021.104506
Received 2 July 2021; Received in revised form 5 September 2021; Accepted 5 October 2021
Available online 30 October 2021

model includes a clustering algorithm to assess the quality of selected features. The wrapper model is very expensive in calculation compared to the filter model (Alelyani et al., 2013). The hybrid method consists of a combination of both methods. In the embedded method, the feature selection method is considered as a part of the learning algorithm (Jović et al., 2015).

In this study, we proposed four different improved versions of the Sine Cosine Algorithm (ImpSCAs) to overcome the deficiencies of the original SCA and to improve the search ability of the SCA. These versions of ImpSCAs are related to the updating mechanism of search agents. Briefly, the main aim of this study is to increase the performance of the original SCA via improving its updating mechanism. We have also integrated some mechanisms such as mutation operator, boundary checking, and simple selection mechanism into the SCA structure. The proposed ImpSCA versions were firstly analyzed in terms of the convergence, search history, trajectory, average distance, and computational complexity. The performances of the ImpSCA versions were evaluated using thirty benchmark functions taken from CEC 2014 test suite (Liang et al., 2014). The binary versions of ImpSCAs were applied for 10 feature selection problems from UCI machine learning repository (Lichman, 2013). The rest of this paper is organized as follows:

A literature review regarding the SCA and feature selection is presented in the Section 2, Section 3 provides brief information about original SCA, and Section 4 presents the improved versions of SCA in detail. The results and discussions are presented in Section 5. Finally, Section 6 includes the conclusions and future works.

2. Literature review

In literature, metaheuristic algorithms are divided into two main groups: population based algorithms and trajectory based algorithms. Tabu Search (Nowicki and Smutnicki, 1996; Gendreau et al., 2006) and Simulated Annealing (Kirkpatrick et al., 1983; Mavridou and Pardalos, 1997) algorithms are well-known examples of the trajectory based metaheuristic algorithms. In these algorithms, the optimization process starts with one solution and it is changed by physical equations during the iterations. In the population based metaheuristic algorithms, optimization process starts with a set of population initialized randomly. Genetic algorithm (GA) (Goldberg, 1989; Holland, 1975), Differential Evolution (DE) algorithm (Storn and Price, 1997; Yüzgeç, 2010), Particle Swarm Optimization (PSO) algorithm (Kennedy and Eberhart, 1995; Poli et al., 2007), Artificial Bee Colony (ABC) algorithm (Karaboga, 2005; Karaboga and Akay, 2009; Karaboga and Basturk, 2008) and Ant Colony Optimization (ACO) algorithm (Dorigo et al., 2006, 1996) are the most popular examples of the population based algorithms. Genetic Algorithm (GA) which was presented by Holland (Holland, 1975), consists of candidate solutions as the chromosomes in the population and the parameters as genes. Particle Swarm Optimization (PSO) algorithm is based on the sociological attitudes of flocking birds (Kennedy and Eberhart, 1995). Differential Evolution (DE), which was introduced by Storn and Price (Storn and Price, 1997) is based on addition of the difference value between two individuals to a third individual in the population. Artificial Bee Colony (ABC) algorithm, which was presented by Karaboga in 2005, is based on the behavior of honey bees. Ant Colony Optimization (ACO) algorithm that is proposed by Dorigo and his friends (Dorigo et al., 2006) imitates the foraging strategies of ants. There are a lot of studies on development of the metaheuristic algorithms and their applications to different optimization problems. Some of these recent metaheuristic algorithms are Firefly algorithm (FA) (Yang, 2010, 2009a), Fruit Fly Optimization algorithm (FOA) (Pan, 2012; Iscan and Gunduz, 2014), Cuckoo Search algorithm (CSA) (Yang and Deb, 2014; Yang, 2009), Touring Ant Colony Optimization algorithm (TACO) (Sreejith et al., 2009), Invasive Weed Optimization algorithm (IWO) (Mehrabian and Lucas, 2006; Zhang et al., 2010), African Vultures Optimization algorithm (AVOA) (Abdollahzadeh et al., 2021a), Artificial Gorilla Troops Optimization algorithm (GTO) (Abdollahzadeh et al., 2021b), Imperialist Competitive algorithm (ICA) (Atashpaz-Gargari and Lucas, 2007; Hosseini and Al Khaled, 2014), Shuffled Frog Leaping algorithm (SFLA) (Eusuff et al., 2006; Bhattacharjee and Sarmah, 2014), Biogeography-Based Optimization algorithm (BBO) (Simon, 2008; Ma and Simon, 2011), Gravitational Search algorithm (GSA) (Rashedi et al., 2009; Sabri et al., 2013), Whale Optimization algorithm (WOA) (Mirjalili and Lewis, 2016; Kaveh and Ghazaan, 2017), Covariance Matrix Adaptation Evolution Strategy (CMA-ES) (Beyer and Sendhoff, 2008; Willjuice Iruthayarajan and Baskar, 2010), Harmony Search algorithm (HSA) (Geem et al., 2001; Yang, 2009b), Cultural Optimization algorithm (COA) (Coello Coello and Becerra, 2004), Gray Wolf Optimization algorithm (GWO) (Mirjalili, 2015a; Mirjalili and Lewis, 2016; Mirjalili et al., 2014), Dragonfly Optimization algorithm (DA) (Mirjalili, 2016a; Sree Ranjini and Murugan, 2017), Grasshopper Optimization algorithm (GOA) (Saremi et al., 2017), Harris Hawk Optimization algorithm (Heidari et al., 2019; Bairathi and Gopalani, 2020), and Moth–Flame Optimizer (MFO) (Mirjalili, 2015b; Xu et al., 2019).

Although a wide range of metaheuristic algorithms is used to solve optimization problems, some difficulties are encountered in using them, such as premature convergence, being stuck local minima, overfitting, etc. Furthermore, the No Free Lunch (NFL) theorem has revealed that any metaheuristic algorithm cannot give good results for all optimization problems (Wolpert and Macready, 1997; Ho and Pepyne, 2002). For these reasons, the researchers focus on introducing new metaheuristic algorithms or developing existing metaheuristic algorithms.

In this work, we interested in the improving update mechanism in the Sine Cosine Algorithm (SCA) introduced by Mirjalili in 2016 (Mirjalili, 2016b). The SCA's update mechanism consists of a mathematical model based on sine and cosine functions. Looking at the advantages of SCA, it is seen that it has a simple mathematical mechanism and few parameters. On the other hand, the SCA tends to be stuck into local minima and premature convergence. In order to overcome these handicaps, there are some studies about various improved SCAs from different perspectives in the literature. Abd Elaziz et al. (2017) proposed an opposition based SCA and better solutions were obtained than the original SCA. To improve the accuracy performance of the SCA algorithm for engineering and global optimization problems, the authors integrated an opposition-based learning mechanism into the SCA structure in their work. (Nenavath and Jatoth, 2018) introduced a hybrid algorithm by combining the differential evolution (DE) algorithm and SCA to solve the problems of global optimization and object tracking. This hybrid method has a better convergence speed than a basic SCA algorithm. In the mainframe of Hybrid SCA-DE, SCA focuses on the exploration and makes a search with big steps to explore the search space, DE emphasizes the exploitation and lets the search agents' move toward the best solution in the promising area. Issa et al. (2018) presented an enhanced version of the adaptive SCA by combining it with a particle swarm optimization (PSO) algorithm. This hybridization process has ensured a good balance between exploration and exploitation stages, thanks to the unique capabilities of PSO and SCA. Qu et al. (2018) proposed an improved SCA based on greedy Levy mutation and neighborhood search to develop the local development ability of the original SCA. It is stated that the proposed strategy in the study can increase the diversity of the search agent in the SCA structure and reduce its oscillation during the search. Suid et al. (2018) focused on increasing the performance of the original SCA. They proposed modification on the random control parameter in the updating mechanism of SCA. The experimental results show that the iSCA has competitive performance compared to the other standard algorithms.

The biggest problem in feature selection is the difficulty in finding the most suitable feature subset due to a large number of possible combinations. In solving a feature selection problem, all possible subsets of features are generated to find the best one. This approach is not a logical way for very large data sets; because if a dataset has k

properties, 2^k solutions should be produced and evaluated, this results in an extremely high computational cost (Iguyon and Elisseeff, 2003). Because of the exponential complexity in this approach, it is impractical for datasets with a very large number of features. A more effective solution for solving feature selection problems is to use a metaheuristic optimization algorithm. Though metaheuristic algorithms do not guarantee to find the best subset of features, an acceptable solution can be generated quickly (Jensen and Shen, 2008). Many researchers have studied the implementation of well-known meta-heuristic algorithms for feature selection problems, such as particle swarm optimization algorithm (Xue et al., 2014; Kothari et al., 2012), differential evolution algorithm (Khushaba et al., 2008; Sikdar et al., 2012), artificial bee colony algorithm (Rao et al., 2019; Shunmugapriya and Kanmani, 2012). Recently, new metaheuristic algorithms inspired by nature have been presented and they have been adapted to solve feature selection problems. Examples of these algorithms are binary dragonfly optimizer (Mafarja et al., 2019), binary grasshopper optimizer (Mafarja et al., 2018), whale optimization algorithm (Mafarja and Mirjalili, 2018b), and hybrid binary ant lion optimizer (Mafarja and Mirjalili, 2018a) proposed by Mafarja et al. chaotic crow search (Sayed et al., 2019) and chaotic salp swarm algorithms (Sayed et al., 2018) proposed by Sayed and his friends.

3. Sine Cosine Algorithm (SCA)

Sine Cosine Algorithm (SCA) was presented by Seyedali Mirjalili in 2016 (Mirjalili, 2016b) as an optimization method based on the sine and cosine mathematical functions. Like other optimization algorithms, the optimization process of SCA includes two stages: exploitation and exploration. The exploration stage, which is the first step of metaheuristic algorithms like SCA, is the process of searching a wide area of search space with a high rate of randomness for finding promising regions. The exploitation stage is the process of visiting those regions of a search space within the neighborhood of formerly visited points. For both stages, the positions of the candidate solutions in SCA are updated by the following equations (Mirjalili, 2016b):

$$X_i^{t+1} = X_i^t + r_1 \times \sin\left(r_2\right) \times \left|r_3 X_b^t - X_i^t\right| \tag{1}$$

$$X_i^{t+1} = X_i^t + r_1 \times \cos\left(r_2\right) \times \left|r_3 X_b^t - X_i^t\right| \tag{2}$$

where X_i^t denotes the position of the current solution, $r_{1,2,3}$ stands for the random numbers, X_b^t represents the position of the best solution, i denotes the dimension, and t is the iteration. Eqs. (1) and (2) are used according to the condition in the following equations:

$$X_i^{t+1} = \begin{cases} X_i^t + r_1 \times \sin\left(r_2\right) \times \left|r_3 X_b^t - X_i^t\right|, r_4 < 0.5 \\ X_i^t + r_1 \times \cos\left(r_2\right) \times \left|r_3 X_b^t - X_i^t\right|, r_4 \geq 0.5 \end{cases} \tag{3}$$

There are four parameters ($r_1 - r_4$) regarding random numbers in the SCA's updating equations given above. The next position's area of a candidate solution is determined by the r_1 parameter as given below:

$$r_1 = a\left(1 - \frac{t}{T}\right) \tag{4}$$

where t is the current iteration number, T is the maximum number of iterations and a is a constant (Mirjalili, 2016b). The r_2 parameter is used to set the movement direction for the next solution. The random weight for X_b^t is found by the r_3 parameter. Lastly, the r_4 parameter provides the switches between the updating mechanisms in Eq. (3). The pseudo code of original SCA is given in Algorithm 1.

Algorithm 1. The Pseudo code of the SCA

```
Initialize a set of the positions of candidate solutions randomly
Calculate the value of the objective function for each candidate
Save the best solution and best candidate's position
While (t < maximum number of iterations)
    Update r1 parameter
    For i = 1 to population size do
        Update random parameters (r2, r3, r4)
        If (r4 < 0.5)
            Update the position of candidate solution using Eq. (1)
        else
            Update the position of candidate solution using Eq. (2)
        End If
        Calculate the value of the objective function for each updated candidate
        Update best solution and its position
    End For i
End While t
Return the best solution at the end of the loop
```

4. Improved Sine Cosine Algorithms (ImpSCAs)

In this section, the proposed improved versions of the original SCA are introduced. The original SCA has some deficiencies such as slow convergence, long run time, getting stuck in local optima, etc. In addition, the updated position of the candidate solution can have worse cost value than its previous cost value and in the exploration stage updated positions exceed the search space are forced to the boundary of the search space. In order to overcome these disadvantages and increase the search performance of the SCA, we focused on update mechanism of the original SCA and developed four different versions of SCA according to the new update mechanisms. In this way, the search ability of the algorithm is improved in the exploration and exploitation phases. The proposed update mechanisms help to locally explore the promising areas in the search space. In addition, three structures added to the original SCA for all of these versions are included in this study. First, a mutation mechanism was implemented to these ImpSCAs to improve the performance of them for the exploitation phase. The mutation mechanism is given below:

$$X_i^{t+1} = X_i^t + 0.1 \times \left(X^{max} - X^{min}\right) \times r_5 \tag{5}$$

where X^{max} stands for the upper boundary, X^{min} denotes the lower boundary of the position of the candidate solution (search agent), and r_5 denotes the normally distributed random number. The second structure is about the boundary checking mechanism of the original SCA. In this modification on the boundary checking mechanism, the candidate solutions' positions that exceed the upper and the lower boundaries of the search space are checked, and they are moved randomly within a maximum of 25% of their boundary values in the search space as given in the equation below:

$$X_i^{t+1} = \begin{cases} X^{min} + 0.25 \times \left(X^{max} - X^{min}\right) \times r_6, \ X_i^{t+1} < X^{min} \\ X^{max} - 0.25 \times \left(X^{max} - X^{min}\right) \times r_6, \ X_i^{t+1} > X^{max} \end{cases} \tag{6}$$

In the original SCA, there is no selection mechanism to select the more suitable candidates for the next iterations. For all versions of ImpSCAs, a simple selection mechanism was added to the original SCA. In this selection method, the updated candidate solution's cost value is compared with the cost value of the previous candidate solution. Then, the candidate with the better cost value is selected for the next iteration. The details of the proposed ImpSCAs are given in sub-sections below.

4.1. Improved Sine Cosine Algorithm 1 (ImpSCA$_1$)

In the first version of ImpSCAs, we introduce the combination of the updating equation with sine function and the updating equation with

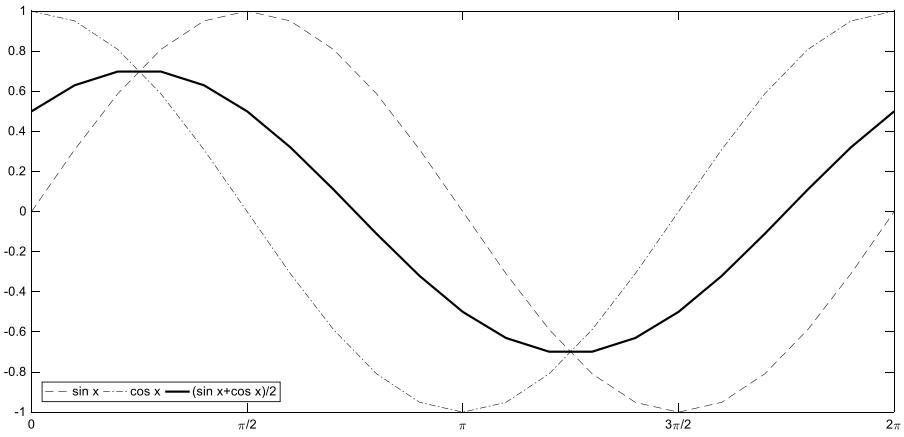


Fig. 1. Sine, Cosine and mean of Sine–Cosine functions with a range of [−1, 1].

cosine function instead of the position updating mechanism (Eqs. (1)–(2)) of the original SCA. The positions of the candidate solutions in the ImpSCA_1 version are updated by the following equation.

$$X_i^{t+1} = X_i^t + r_1 \times 0.5 \times \left[\sin\left(r_2\right) + \cos(r_2)\right] \times \left|r_3 X_b^t - X_i^t\right| \tag{7}$$

Here, X_b^t stands for the global best, which also is the destination solution, among all X_i^t solution positions. In this proposed method, the mean of Sine and Cosine functions for the random parameter (r_2) is used in the position updating. Fig. 1 shows the Sine, Cosine and mean of Sine–Cosine functions in a range of [−1, 1]. As can be seen from Fig. 1, the curve of the mean of the Sine and Cosine varies in the range of [−0.75, 0.75]. The positions of candidate solutions are effectively changed by the proposed position update method of the ImpSCA_1 version.

4.2. Improved Sine Cosine Algorithm 2 (ImpSCA_2)

In the second version of ImpSCAs (ImpSCA_2), the position of a candidate solution selected randomly from the population is used instead of the best candidate solution's position (destination position). The equation of updating position used in ImpSCA_2 is given below:

$$X_i^{t+1} = X_i^t + r_1 \times 0.5 \times \left[\sin\left(r_2\right) + \cos(r_2)\right] \times \left|r_3 X_r^t - X_i^t\right| \tag{8}$$

where X_r^t denotes the position of the candidate solution randomly selected from the population. In the original SCA, the global best solution cannot be changed during the long iterations when stuck at local optima. Therefore, after certain iteration, all candidate solutions are gathered very close to the position of the global best solution, and this causes late convergence or failure to recover from the local optima. In the proposed updating position method of ImpSCA_2, using X_r^t instead of X_b^t has led to improving the diversity of the population and it provides to discover based on the random information regarding the search space.

4.3. Improved Sine Cosine Algorithm 3 (ImpSCA_3)

The main idea of ImpSCA_3 is based on the weighted average of the updated positions found by the update equations used in the original SCA. The proposed update mechanism is given below:

$$X_1 = X_i^t + r_1 \times \sin\left(r_2\right) \times \left|r_3 X_b^t - X_i^t\right| \tag{9}$$

$$X_2 = X_i^t + r_1 \times \cos\left(r_2\right) \times \left|r_3 X_b^t - X_i^t\right| \tag{10}$$

$$X_i^{t+1} = 0.5 \times (r_4 X_1 + r_5 X_2) \tag{11}$$

where X_1 and X_2 represent the updated positions with sine and cosine respectively, r_4, r_5 denote the normally distributed random number. In the original SCA, which position updating mechanism is used is determined according to the condition in the Eq. (3). In this proposed version, this condition was removed, and both of the update equations were taken, then the weighted average of the updated position values (X_1 and X_2) was used.

4.4. Improved Sine Cosine Algorithm 4 (ImpSCA_4)

For the last version of ImpSCAs (ImpSCA_4), it was inspired by the crossover operator in the genetic algorithm. The crossover operator uses the updated position values (X_1 and X_2) calculated by the updating mechanisms of the positions of the candidate solutions, which are used in the previous version of ImpSCA. At the end of the crossover operation, two new candidate solutions ($\acute{X}_1, \acute{X}_2$) are obtained, and the position of the next solution is the average of these new candidates. The equations of the updating mechanism of the solution's positions are given by:

$$\acute{X}_1 = \alpha X_1 + (1-\alpha) X_2 \tag{12}$$

$$\acute{X}_2 = (1-\alpha) X_1 + \alpha X_2 \tag{13}$$

$$X_i^{t+1} = 0.5 \times (\acute{X}_1 + \acute{X}_2) \tag{14}$$

where α stands for the crossover control parameter in a range of [0,1] and $\acute{X}_1, \acute{X}_2$ denotes the new individuals or new candidate solutions. This proposed crossover operator helps to improve the exploitative ability of the ImpSCA_4. The pseudo codes of all ImpSCA versions are given in Algorithm 2. The italic lines in this pseudo code show the new parts added to it.

5. Results and discussions

In this section, first of all, we have analyzed the proposed ImpSCAs in terms of convergence, search history, trajectory, average distance metrics, and computational complexity. These analyses are described in detail together with the selected four benchmark problems in Section 5.1. The initial search agents are equal to each other in SCA and ImpSCAs for all analyses. Secondly, the proposed ImpSCAs were evaluated on thirty benchmark problems based on the IEEE Congress on Evolutionary Computation (CEC) 2014 test suite (Liang et al., 2014). The proposed ImpSCAs were compared with the original SCA for the benchmark tests with dimension sizes of 10D, 30D and 50D. In the last sub-section, we adapted SCA and ImpSCA versions for feature selection problems and these versions were tested for ten feature selection datasets taken from the UCI data repository (Lichman, 2013).

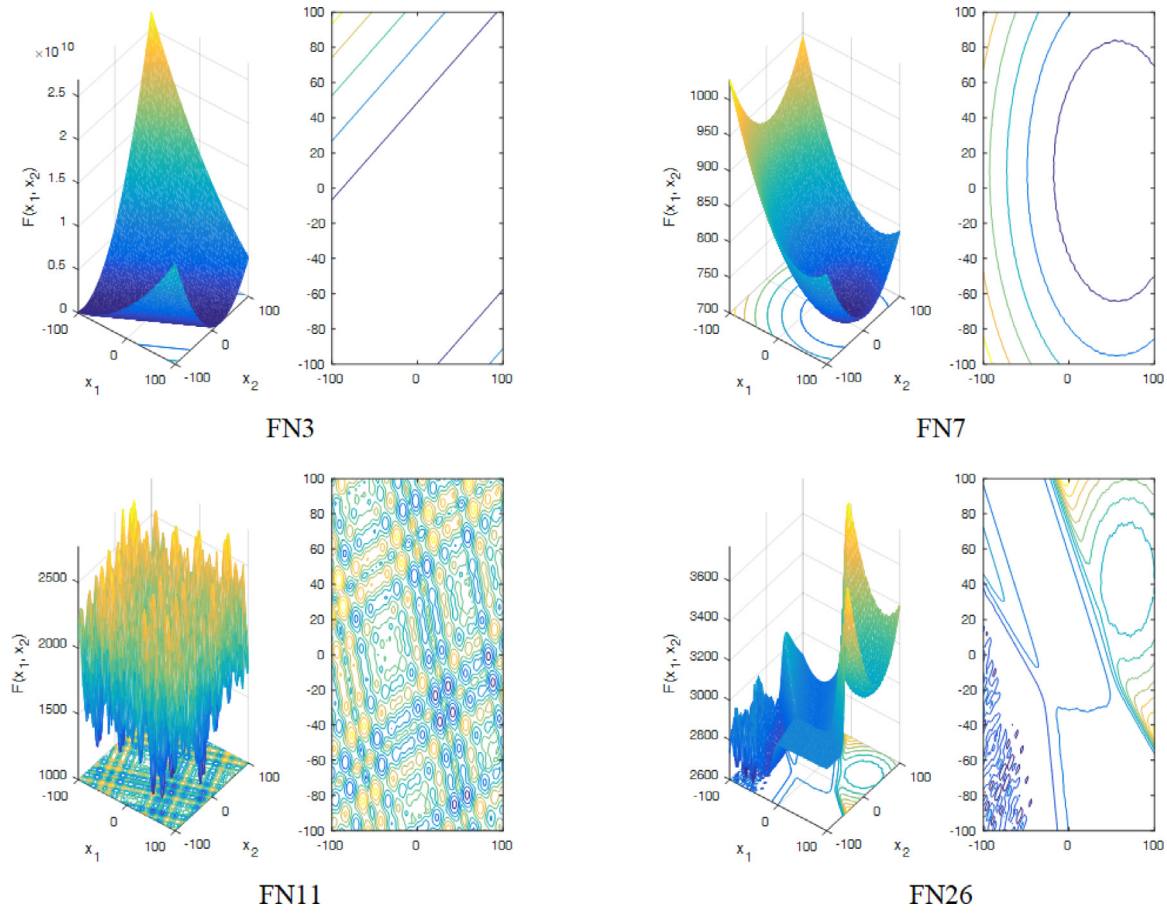


Fig. 2. Unimodal, multimodal and composition benchmark functions used for analyzing the proposed ImpSCAs.

Table 1
CEC 2014 benchmark functions.

No.	Function name	Solution	No.	Function name	Solution
FN1	Rotated High Conditioned Elliptic	100	FN16	Shifted and Rotated Expanded Scaffer's F6	1600
FN2	Rotated Bent Cigar	200	FN17	Hybrid Function 1 (N = 3)	1700
FN3	Rotated Discus	300	FN18	Hybrid Function 2 (N = 3)	1800
FN4	Shifted and Rotated Rosenbrock	400	FN19	Hybrid Function 3 (N = 4)	1900
FN5	Shifted and Rotated Ackley	500	FN20	Hybrid Function 4 (N = 4)	2000
FN6	Shifted and Rotated Weierstrass	600	FN21	Hybrid Function 5 (N = 5)	2100
FN7	Shifted and Rotated Griewank	700	FN22	Hybrid Function 6 (N = 5)	2200
FN8	Shifted Rastrigin	800	FN23	Composition Function 1 (N = 5)	2300
FN9	Shifted and Rotated Rastrigin	900	FN24	Composition Function 2 (N = 3)	2400
FN10	Shifted Schwefel	1000	FN25	Composition Function 3 (N = 3)	2500
FN11	Shifted and Rotated Schwefel	1100	FN26	Composition Function 4 (N = 5)	2600
FN12	Shifted and Rotated Katsuura	1200	FN27	Composition Function 5 (N = 5)	2700
FN13	Shifted and Rotated HappyCat	1300	FN28	Composition Function 6 (N = 5)	2800
FN14	Shifted and Rotated GHBat	1400	FN29	Composition Function 7 (N = 3)	2900
FN15	Expanded F4 plus F7	1500	FN30	Composition Function 8 (N = 3)	3000

5.1. Analyses of the proposed ImpSCAs

CEC 2014 test suite consists of unimodal, simple multimodal, hybrid, and composition functions. The benchmark functions and their names are given in Table 1. The detailed information about CEC 2014 test suite can be seen from the study of Liang and his friends (Liang et al., 2014). In this study, we determined four benchmark problems from the CEC 2014 test suite to show the analyses' results of proposed ImpSCAs.

The first function (FN3) is the rotated discus function from unimodal functions. The second function (FN7) is the shifted and rotated Griewank's function from simple multimodal functions. The third function (FN11) is the shifted and rotated Schwefel's function from simple multimodal functions. As last, the fourth function (FN26) is the composition function 4 (N = 5) from composition functions. 3-*D* maps and contour lines for 2-*D* of these functions are shown in Fig. 2.

5.1.1. Convergence analysis

The convergence behavior of a metaheuristic algorithm gives some important information on its performance. In this analysis, the proposed ImpSCA versions and the original SCA were compared with respect to their function error values. We utilized uniform random initialization within the search space. The same initial populations were

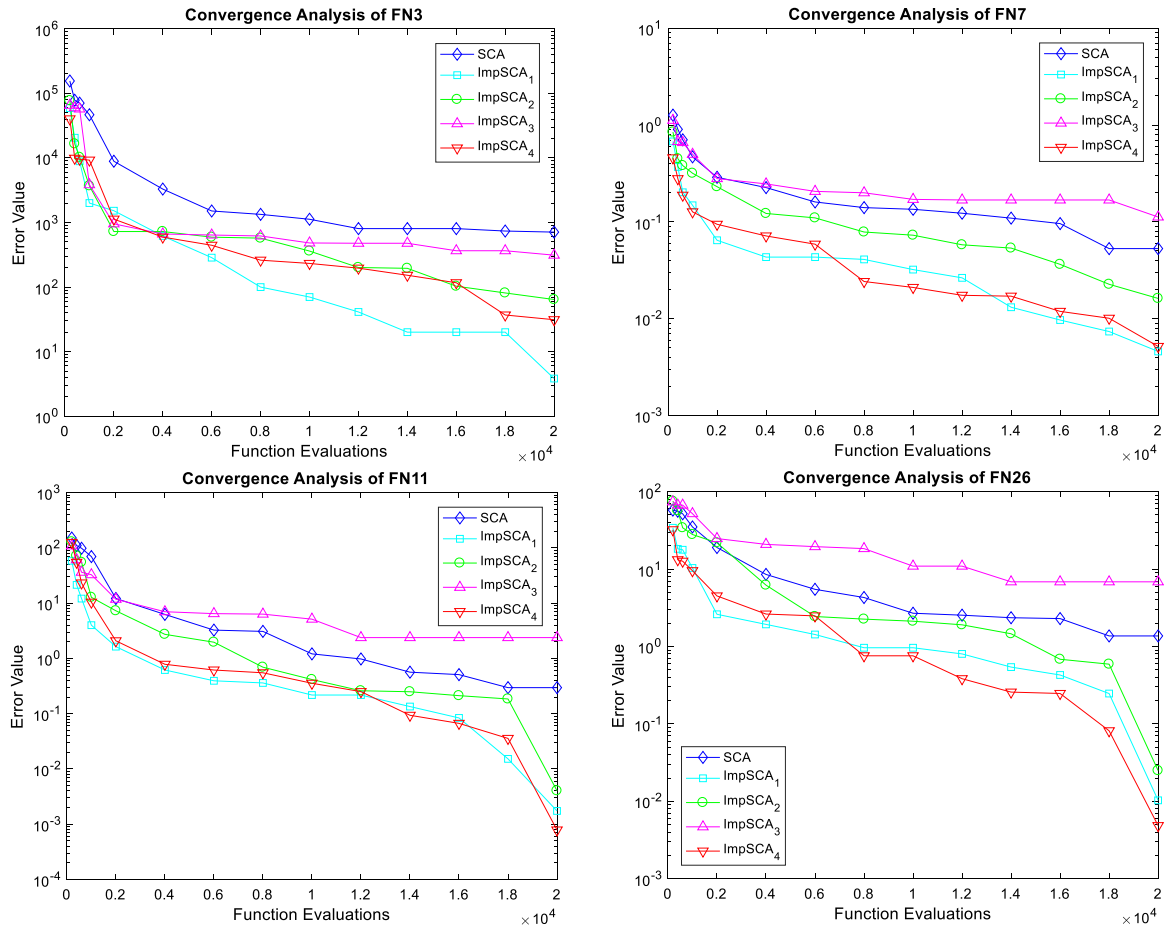


Fig. 3. Convergence analyses of ImpSCAs.

Algorithm 2. The Pseudo code of the ImpSCA versions

Initialize a set of the positions of candidate solutions randomly
Calculate the value of the objective function for each candidate
Save the best solution and best candidate's position
While (t < maximum number of iterations)
    old candidates = *current candidates*
    Update r_1 parameter
    For i =1 **to** population size **do**
        Update random parameters (r_2, r_3)
        - *Update the position of candidate solution using Eq. (7) in ImpSCA1*
        - *Update the position of candidate solution using Eq. (8) in ImpSCA2*
        - *Update the position of candidate solution using Eqs. (9,10,11) in ImpSCA3*
        - *Update the position of candidate solution using Eqs. (9,10,12,13,14) in ImpSCA4*

        Update the positions using mutation operator (Eq.5)
        Send the candidate into the search space using the boundary checking mechanism (Eq. 6)
        Calculate the value of the objective function for each updated candidate
        If *(cost value of updated candidate < cost value of old candidate)*
            next candidate = *updated candidate*
        Else
            next candidate = *old candidate*
        End If
        Update best solution and its position
    End For i
End While t
Return the best solution at the end of the loop

taken for four benchmark functions. The results of the convergence analyses for these functions are shown in Fig. 3. These curves give the average fitness for 5 runs of the elite solutions found by the proposed ImpSCAs and SCA during the optimization.

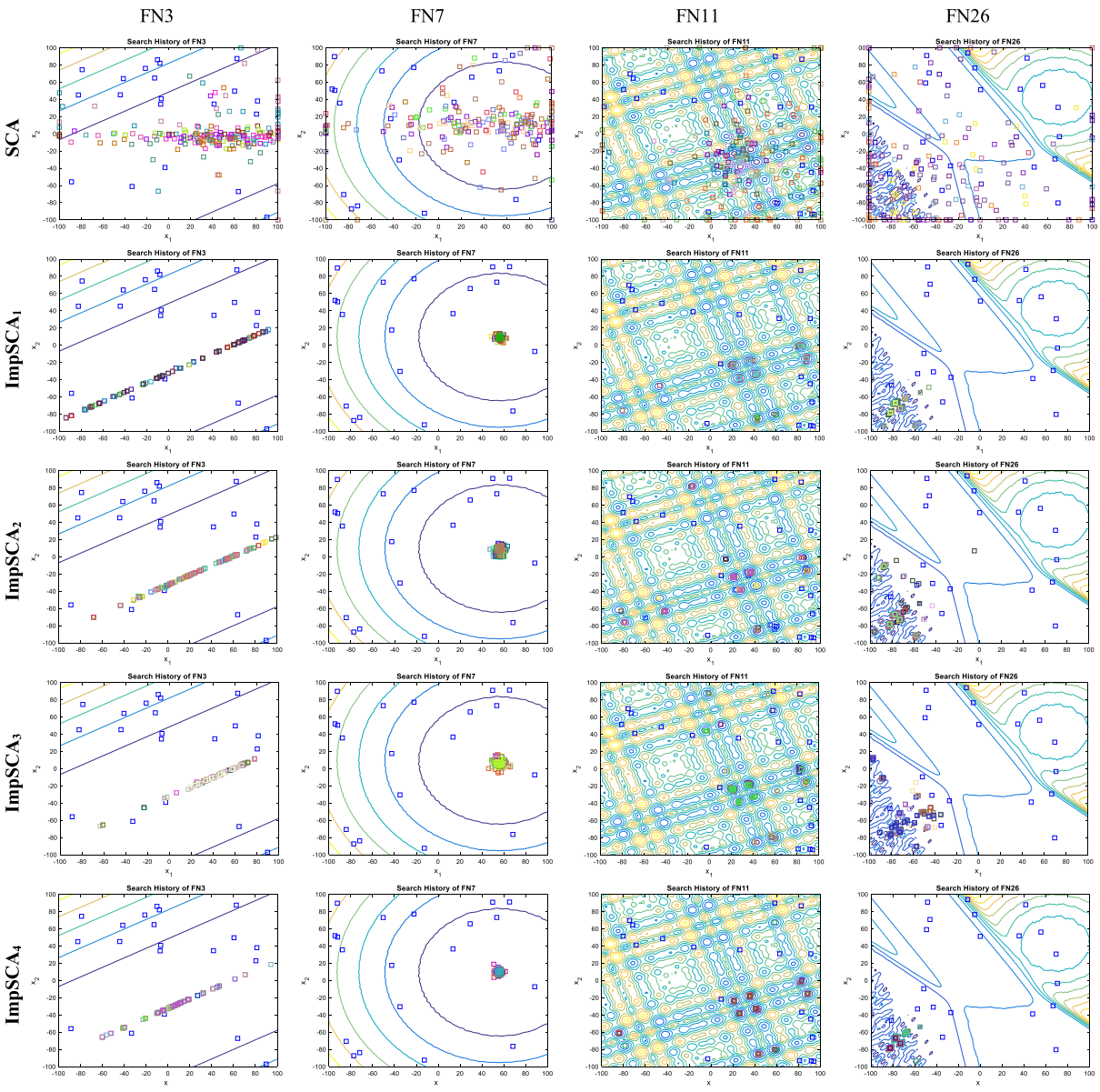


Fig. 4. Search history analysis of ImpSCAs (each color denotes the search agents in the updated population for every 100 steps of the iteration). . (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

In these figures, the convergence curves of ImpSCAs and original SCA were drawn on a logarithmic scale. The decreasing trend in the convergence curves of ImpSCA versions on the four benchmark functions is better than that of the original SCA. This shows the ability of the proposed ImpSCAs to find a solution closer to the global optimum. As can be seen from the figure of all functions except FN3, the SCA only performs better than that of $ImpSCA_3$.

5.1.2. Search history analysis

Search history analysis gives information about how the search agents or the candidate solutions move in the search space during the optimization process. Fig. 4 shows the search history of the search agents for selected benchmark functions. In the search analysis of each benchmark function, the initial population is taken the same for all algorithms and is shown in blue in these subfigures. The search agents in the updated population are shown in random colors for every 100 steps of the iteration. The number of the search agent is 20. In this figure, the original SCA partially continues exploration by not always focusing on solution positions.

However, the versions of ImpSCA explore the regions close to global optima in the search space. Looking at the result of the search history analyses obtained by ImpSCAs, the distribution of candidate solution points around the global optima is higher than that by SCA, this shows that the ImpSCA versions search the most promising area of the search space in the exploration and exploitation phases. The reason for the high success of ImpSCA versions in search history analysis is not only the innovations in the update mechanisms, but also the mutation operator, boundary checking mechanism, and simple selection mechanism added to the algorithm.

5.1.3. Trajectory analysis

Trajectory analysis represents how the position of the elite search agent or the best candidate solution changes in the search space during iterations. For selected benchmark functions, trajectory analysis results are shown in Figs. 5 and 6. In the first figure, changes in the position of the elite search agent on the contour surface of the search area are given for original SCA and ImpSCAs. The red markers indicate the positions of search agents found by SCA and ImpSCAs at the end of the optimization. In Fig. 6, the change in the position of the elite search

Table 2
10D CEC2014 benchmark results for SCA and ImpSCA versions.

		FN1	FN2	FN3	FN4	FN5	FN6	FN7	FN8	FN9	FN10	FN11	FN12	FN13	FN14	FN15
SCA	*Best*	2.04e6	1.35e8	1.32e3	32.74	20.17	3.82	4.25	24.25	19.44	562.47	502.42	0.61	0.34	0.36	3.60
	Worst	1.97e7	1.10e9	8.24e3	81.27	20.51	7.97	15.35	54.56	53.17	1.25e3	1.69e3	1.51	0.98	1.43	11.22
	Mean	6.15e6	4.67e8	2.71e3	51.57	20.34	6.08	8.45	35.59	38.28	905.87	1.18e3	1.13	0.55	0.80	6.72
	Median	6.12e6	4.57e8	2.15e3	50.49	20.35	6.00	8.50	35.78	37.02	877.84	1.22e3	1.17	0.53	0.84	6.38
	Std	2.86e6	1.92e8	1.48e3	9.20	**0.08**	0.97	2.32	6.06	7.37	177.22	221.83	0.20	0.12	0.28	1.40
$ImpSCA_1$	*Best*	**2.82e4**	2.30e5	**353.46**	1.02	**6.25**	0.97	0.53	3.21	5.36	22.08	**35.15**	0.27	0.12	**0.08**	0.93
	Worst	1.79e6	1.51e6	5.86e3	**35.70**	20.41	**2.06**	**0.95**	9.74	**13.10**	**424.45**	**532.63**	1.04	0.29	**0.23**	2.35
	Mean	**3.11e5**	**6.00e5**	**1.91e3**	**9.39**	19.60	**1.36**	0.75	**5.82**	8.40	**115.90**	**305.01**	0.60	**0.19**	0.17	1.54
	Median	**2.52e5**	**5.68e5**	**1.38e3**	**8.06**	20.30	**1.31**	**0.75**	**5.76**	8.23	**69.45**	**324.78**	0.58	**0.19**	0.17	1.51
	Std	2.73e5	2.59e5	1.31e3	**6.97**	2.48	**0.28**	**0.10**	**1.27**	**1.87**	91.83	128.52	0.14	**0.03**	**0.03**	0.33
$ImpSCA_2$	*Best*	8.40e5	1.58e6	1.42e3	2.43	13.56	2.20	0.54	9.85	13.98	139.61	195.11	**0.22**	0.17	0.14	1.79
	Worst	6.83e6	3.21e7	**5.42e3**	38.88	**20.36**	4.19	1.98	27.91	26.46	470.09	568.43	**0.67**	0.36	0.31	3.62
	Mean	3.64e6	6.75e6	3.27e3	20.44	**19.47**	3.44	1.36	19.40	20.99	268.11	379.83	**0.43**	0.28	0.24	2.87
	Median	3.64e6	5.11e6	3.30e3	18.99	20.23	3.49	1.33	19.53	21.19	265.94	365.51	**0.42**	0.28	0.25	2.90
	Std	1.26e6	4.90e6	**961.05**	8.88	1.83	0.49	0.27	3.33	2.73	**62.97**	**87.68**	**0.10**	0.04	**0.03**	0.41
$ImpSCA_3$	*Best*	1.10e7	7.35e8	2.49e3	103.37	20.06	6.75	18.05	40.57	40.83	439.45	816.35	0.59	0.76	3.00	7.91
	Worst	3.15e7	3.84e9	1.11e4	572.40	20.48	8.79	84.08	70.63	59.66	1.10e3	1.37e3	1.65	2.01	10.49	25.26
	Mean	1.92e7	2.81e9	5.98e3	333.32	20.33	7.85	61.02	58.29	51.43	791.35	1.14e3	1.14	1.32	6.92	15.88
	Median	1.14e7	1.51e9	3.19e3	151.93	**20.15**	7.17	27.93	45.94	43.41	559.12	906.19	0.72	0.78	3.76	10.54
	Std	4.13e6	5.51e8	1.53e3	111.15	**0.08**	0.49	15.28	6.00	4.25	137.04	127.05	0.20	0.30	1.57	3.73
$ImpSCA_4$	*Best*	4.81e4	**1.66e5**	458.10	**0.55**	10.27	**0.72**	**0.47**	**1.79**	**3.76**	**13.48**	68.81	0.37	**0.10**	**0.08**	**0.87**
	Worst	**1.43e6**	**1.35e6**	9.14e3	37.77	20.38	2.67	0.98	**8.77**	16.91	470.27	622.23	0.94	**0.27**	0.28	**2.00**
	Mean	3.75e5	6.15e5	2.34e3	10.90	19.98	1.45	**0.74**	6.04	**8.10**	118.24	336.02	0.58	**0.19**	**0.16**	**1.47**
	Median	3.24e5	6.32e5	1.41e3	8.37	20.27	1.43	**0.75**	6.13	**8.07**	81.50	331.22	0.56	**0.19**	**0.15**	**1.50**
	Std	**2.68e5**	**2.49e5**	1.98e3	8.40	1.55	0.39	**0.10**	1.44	2.42	98.96	146.65	0.12	0.04	0.04	**0.25**

		FN16	FN17	FN18	FN19	FN20	FN21	FN22	FN23	FN24	FN25	FN26	FN27	FN28	FN29	FN30
SCA	*Best*	2.76	3.78e3	2.09e3	3.47	117.95	1.51e3	32.63	333.00	138.29	152.10	100.35	9.88	407.02	490.29	715.39
	Worst	3.77	4.36e4	6.25e4	6.34	4.74e3	1.80e4	70.30	344.44	164.68	203.95	100.75	410.70	533.47	1.63e4	2.27e3
	Mean	3.19	1.57e4	1.49e4	4.95	1.09e3	6.20e3	53.58	339.15	150.11	195.30	100.54	326.98	455.45	3.45e3	1.30e3
	Median	3.22	1.10e4	1.23e4	4.88	477.99	5.50e3	54.05	339.17	150.28	202.12	100.54	402.83	438.97	2.92e3	1.20e3
	Std	0.23	1.15e4	1.17e4	0.69	1.31e3	3.85e3	9.22	2.87	5.33	15.15	0.10	156.46	44.33	2.65e3	380.48
$ImpSCA_1$	*Best*	1.56	728.23	141.37	**1.19**	**51.89**	**345.46**	**22.98**	329.51	108.28	**112.96**	**100.08**	**2.57**	205.62	288.71	**557.99**
	Worst	**2.92**	**5.47e3**	**5.18e3**	**2.17**	**1.25e3**	**3.15e3**	35.39	329.97	**118.25**	201.51	**100.23**	400.54	494.44	577.60	**1.38e3**
	Mean	2.30	2.03e3	**1.15e3**	**1.87**	258.05	**1.04e3**	26.76	329.65	**113.62**	**134.33**	**100.18**	64.61	399.04	390.16	**860.63**
	Median	2.34	1.90e3	**444.58**	**1.89**	215.24	**944.00**	26.60	329.62	113.68	**126.65**	100.18	**3.89**	380.37	**381.60**	**811.75**
	Std	0.31	964.25	1.42e3	0.19	**204.56**	**496.80**	2.14	0.08	**2.12**	21.56	0.03	141.87	50.96	65.79	**178.72**
$ImpSCA_2$	*Best*	2.18	1.52e3	175.80	1.99	96.39	705.70	30.22	**200.00**	122.62	142.67	100.12	4.64	**190.76**	300.70	1.12e3
	Worst	3.10	6.35e3	5.22e3	3.89	2.93e3	4.87e3	56.05	200.03	149.44	**191.70**	100.31	**11.06**	514.46	703.67	2.36e3
	Mean	2.80	2.82e3	1.92e3	2.78	1.13e3	2.09e3	40.11	**200.00**	135.33	174.18	100.23	**7.55**	**340.90**	413.77	1.85e3
	Median	2.85	2.58e3	1.52e3	2.74	883.28	1.90e3	39.61	**200.00**	136.87	176.50	100.23	7.56	**351.77**	404.38	1.87e3
	Std	0.20	943.35	**1.22e3**	0.37	800.87	882.99	5.79	**0.00**	6.50	**11.81**	0.03	**1.63**	88.64	78.38	246.60
$ImpSCA_3$	*Best*	3.08	8.63e3	6.49e3	5.30	238.77	3.72e3	37.20	**200.00**	156.27	177.59	100.81	32.29	200.00	7.94e3	2.49e3
	Worst	3.71	3.06e5	1.51e4	9.65	6.65e3	1.76e4	185.89	**200.00**	200.00	200.92	102.43	412.48	1.23e3	1.06e6	7.43e3
	Mean	3.47	1.08e5	1.26e4	7.32	4.00e3	9.55e3	93.19	**200.00**	170.79	189.26	101.98	98.35	923.77	2.05e5	3.67e3
	Median	3.10	1.28e4	7.47e3	5.38	1.25e3	3.76e3	52.01	**200.00**	157.13	192.14	101.05	46.82	666.47	1.38e4	2.50e3
	Std	**0.15**	6.93e4	1.43e3	0.99	1.27e3	3.35e3	33.14	**0.00**	8.54	14.56	0.33	54.05	157.05	2.02e5	932.42
$ImpSCA_4$	*Best*	**1.17**	**673.80**	**87.34**	1.56	70.02	404.92	23.47	329.53	**107.05**	116.74	100.12	2.67	361.51	**257.57**	594.51
	Worst	**2.92**	6.61e3	7.20e3	2.21	2.25e3	4.22e3	**30.78**	329.83	121.05	201.57	**100.23**	400.52	**492.45**	**539.73**	1.39e3
	Mean	**2.21**	**1.97e3**	1.43e3	1.94	**251.68**	1.21e3	**26.47**	329.64	113.75	142.18	**100.18**	48.09	398.96	**388.23**	889.34
	Median	**2.19**	**1.88e3**	768.81	1.94	**191.85**	975.76	**26.49**	329.63	**113.66**	129.75	**100.17**	4.01	381.25	389.57	820.41
	Std	0.40	**922.99**	1.70e3	**0.14**	312.80	805.20	**1.52**	0.07	2.73	26.97	**0.02**	121.16	**38.87**	**53.93**	185.73

agent is illustrated separately for two dimensions. SCA is represented by blue solid line, $ImpSCA_1$ is shown in cyan color, green color indicates $ImpSCA_2$, magenta solid line indicates $ImpSCA_3$, and $ImpSCA_4$ is shown in red solid line.

It is clearly observed that the elite search agent's position is faster updated in the exploration phase and the elite search agent gets closer to global optima in the exploitation phase. But there are some differences between trajectory analysis results of SCA and ImpSCAs. The original SCA cannot find an elite solution close to the global optima at the end of the iterations. For example, it is seen from Fig. 6 that the elite search agent of SCA gets stuck in the local minima for FN11 function. The proposed ImpSCA versions explore more successfully search space and find elite positions closer to the global optima in both phases.

5.1.4. Average distance analysis

In this analysis, we examined the average distance of first search agent to others during optimization. The average distance analysis shows the exploratory or exploitative search behaviors of the proposed ImpSCAs. Fig. 7 shows the average distance analyses of SCA and ImpSCAs for selected benchmark functions.

To calculate the average distance ($\overline{d}$) between the first dimensions of the first agent and of the rest agents, the following equation is utilized:

$$\overline{d} = \frac{1}{N-1}\sum_{i=2}^{N}\left|X_1 - X_i\right| \tag{15}$$

where N denotes the population size, X_1 represents the first agents, and X_i stands for i_{th} agent in the population. It can be seen from figure that the average distance obtained by the original SCA shows a lot of oscillation and fluctuations for benchmark functions. Thanks to the selection mechanism added into the ImpSCAs, less oscillations and fluctuations were observed in the result graphs.

Table 3
30D CEC2014 benchmark results for SCA and ImpSCA versions.

		FN1	FN2	FN3	FN4	FN5	FN6	FN7	FN8	FN9	FN10	FN11	FN12	FN13	FN14	FN15
	Best	9.89e7	9.31e9	2.70e4	660.31	20.83	29.50	81.22	168.48	221.38	4.75e3	6.25e3	1.90	1.70	27.74	290.95
	Worst	4.15e8	2.22e10	6.19e4	1.42e3	21.02	39.04	210.36	290.02	307.00	6.80e3	7.58e3	3.01	3.49	53.60	2.09e4
SCA	*Mean*	2.24e8	1.6e10	3.79e4	1.02e3	20.94	34.42	133.10	235.63	266.53	5.88e3	7.04e3	2.45	2.89	41.09	2.82e3
	Median	2.13e8	1.6e10	3.60e4	994.60	20.95	34.31	130.93	236.20	270.23	5.87e3	7.07e3	2.44	2.85	41.17	1.47e3
	Std	7.18e7	3.12e9	7.28e3	204.17	**0.05**	2.61	29.10	20.84	20.38	450.54	**276.52**	0.23	0.33	5.51	3.62e3
	Best	**9.44e6**	**1.57e8**	1.58e4	141.75	**20.68**	**12.95**	2.96	**65.15**	**87.67**	**2.32e3**	3.69e3	1.04	**0.28**	**0.18**	**14.62**
	Worst	4.03e7	**3.52e8**	**4.11e4**	**205.16**	20.99	21.24	**5.41**	**104.81**	138.92	4.36e3	5.47e3	2.05	0.51	0.32	21.53
$ImpSCA_1$	*Mean*	**2.09e7**	**2.37e8**	**2.84e4**	**177.06**	20.91	**16.73**	**3.75**	**88.67**	**116.27**	3.40e3	4.70e3	1.65	0.40	**0.26**	18.22
	Median	**2.05e7**	**2.33e8**	**2.84e4**	**177.75**	20.93	**16.88**	**3.70**	90.78	**115.21**	3.44e3	4.76e3	1.65	0.41	**0.27**	18.36
	Std	7.17e6	4.37e7	6.77e3	**14.52**	0.07	1.83	**0.54**	**9.55**	12.02	429.50	415.63	0.24	0.05	**0.03**	1.50
	Best	2.56e7	2.11e9	3.57e4	385.68	20.71	20.33	27.21	129.33	151.31	2.75e3	**3.23e3**	**0.95**	0.50	11.14	60.34
	Worst	1.15e8	5.03e9	5.52e4	610.01	**20.97**	24.93	47.30	166.90	195.07	**3.67e3**	**4.73e3**	**1.78**	0.77	21.41	132.23
$ImpSCA_2$	*Mean*	8.54e7	3.92e9	4.79e4	496.72	**20.88**	23.12	37.77	149.66	179.41	**3.23e3**	**4.28e3**	**1.42**	0.65	16.60	92.29
	Median	8.67e7	3.85e9	4.79e4	500.66	20.89	23.16	37.63	149.75	181.22	**3.25e3**	**4.36e3**	**1.45**	0.66	16.41	93.15
	Std	1.62e7	6.36e8	4.00e3	50.88	0.07	1.14	5.06	9.57	9.53	**203.46**	321.44	0.21	0.06	2.44	16.04
	Best	5.00e8	4.16e10	5.31e4	3.58e3	20.76	32.95	374.15	281.16	268.70	5.14e3	6.16e3	1.98	5.34	123.45	2.45e4
	Worst	8.26e8	5.72e10	7.01e4	6.20e3	21.02	37.15	519.71	325.85	333.80	6.59e3	7.43e3	2.91	6.46	183.53	8.72e4
$ImpSCA_3$	*Mean*	6.63e8	5.00e10	6.10e4	4.50e3	20.95	35.45	453.20	302.41	311.15	6.05e3	6.92e3	2.51	5.90	154.68	5.52e4
	Median	5.32e8	4.26e10	5.36e4	3.66e3	**20.83**	32.98	378.82	281.49	281.15	5.19e3	6.29e3	2.15	5.42	130.06	2.53e4
	Std	7.60e7	3.59e9	**3.97e3**	601.72	**0.05**	**1.07**	35.89	11.66	14.89	337.47	311.77	**0.20**	0.25	12.50	1.58e4
	Best	1.01e7	1.70e8	**1.43e4**	**132.00**	20.73	13.19	**2.53**	66.17	96.62	2.33e3	3.45e3	1.18	0.29	0.20	14.68
	Worst	**3.53e7**	3.55e8	4.64e4	211.97	20.98	**20.76**	6.28	106.48	**136.20**	4.39e3	5.51e3	2.13	**0.49**	**0.31**	**20.90**
$ImpSCA_4$	*Mean*	2.30e7	2.41e8	3.01e4	177.57	20.90	16.98	3.89	90.04	117.23	3.45e3	4.80e3	1.69	**0.39**	**0.26**	**18.02**
	Median	2.24e7	2.39e8	3.04e4	179.19	20.91	17.04	3.84	**89.53**	118.13	3.49e3	4.85e3	1.72	**0.39**	**0.27**	**18.01**
	Std	**5.65e6**	**4.18e7**	7.27e3	18.51	0.06	2.03	0.72	9.91	**8.98**	491.17	358.18	0.23	**0.04**	**0.03**	**1.49**
		FN16	FN17	FN18	FN19	FN20	FN21	FN22	FN23	FN24	FN25	FN26	FN27	FN28	FN29	FN30
	Best	11.69	1.20e6	6.25e7	47.67	4.87e3	4.11e5	493.49	347.61	200.13	215.44	101.17	477.35	1.59e3	4.33e6	8.40e4
	Worst	13.29	1.60e7	4.27e8	150.31	2.62e4	4.32e6	992.69	426.78	204.98	241.38	103.34	1.32e3	2.78e3	3.27e7	4.62e5
SCA	*Mean*	12.79	6.61e6	1.85e8	90.76	1.31e4	1.48e6	754.43	370.06	200.99	227.26	102.33	722.12	2.00e3	1.34e7	2.48e5
	Median	12.84	6.10e6	1.81e8	89.10	1.21e4	1.29e6	753.11	368.02	200.58	226.22	102.48	566.61	1.87e3	1.09e7	2.39e5
	Std	0.31	2.94e6	8.68e7	24.23	4.83e3	8.55e5	125.39	13.70	0.91	6.18	0.56	291.13	317.48	7.63e6	8.39e4
	Best	11.04	**7.33e4**	**1.08e5**	**12.46**	**1.40e3**	**1.73e4**	59.27	320.34	200.08	207.37	100.35	302.06	1.06e3	1.56e4	1.15e4
	Worst	12.44	1.18e6	1.09e6	**24.12**	**1.19e4**	**2.07e5**	301.21	323.46	200.56	213.38	**100.52**	643.76	1.34e3	**5.24e4**	**3.11e4**
$ImpSCA_1$	*Mean*	**11.81**	4.29e5	4.43e5	16.64	**5.61e3**	**8.29e4**	**140.24**	321.77	200.20	210.83	**100.44**	**444.22**	1.16e3	2.79e4	**1.93e4**
	Median	**11.83**	3.74e5	4.28e5	16.29	5.37e3	**7.17e4**	**122.21**	321.64	200.18	210.89	**100.44**	**422.40**	**1.15e3**	2.73e4	1.87e4
	Std	0.28	2.41e5	2.27e5	2.37	2.47e3	**4.30e4**	61.38	0.79	0.10	1.33	**0.04**	**53.45**	50.08	**6.59e3**	**4.99e3**
	Best	11.45	2.02e5	9.23e5	44.46	1.08e4	7.25e4	197.83	335.69	200.05	**200.00**	101.27	**113.78**	2.13e3	3.30e4	4.95e4
	Worst	**12.29**	1.74e6	1.65e7	98.16	3.05e4	2.94e5	402.74	357.81	200.10	**200.00**	105.48	**526.04**	3.64e3	1.10e6	2.34e5
$ImpSCA_2$	*Mean*	11.95	8.91e5	5.45e6	78.90	2.19e4	1.57e5	314.70	347.28	200.09	**200.00**	103.74	470.81	2.96e3	3.53e5	1.25e5
	Median	11.99	8.93e5	4.37e6	79.99	2.21e4	1.60e5	316.03	347.21	200.09	**200.00**	103.77	480.66	2.99e3	3.08e5	1.22e5
	Std	0.20	3.57e5	3.70e6	13.04	4.59e3	4.79e4	51.43	5.21	0.01	**0.00**	0.71	67.14	359.93	2.09e5	4.23e4
	Best	12.49	5.65e6	1.29e8	137.81	1.74e4	1.86e6	841.68	**200.00**	**200.00**	**200.00**	106.60	647.34	**200.00**	**200.00**	7.40e5
	Worst	13.06	4.10e7	5.14e8	276.09	9.06e4	1.04e7	1.71e3	**200.00**	**200.00**	**200.00**	153.90	990.48	7.06e3	4.03e8	2.50e6
$ImpSCA_3$	*Mean*	12.84	2.39e7	3.16e8	173.90	5.00e4	5.56e6	1.40e3	**200.00**	**200.00**	**200.00**	130.25	821.69	5.30e3	1.36e8	1.53e6
	Median	12.53	7.04e6	1.61e8	142.55	1.97e4	2.14e6	990.29	**200.00**	**200.00**	**200.00**	106.95	648.92	3.25e3	1.57e8	7.77e5
	Std	**0.15**	7.99e6	9.22e7	22.70	1.53e4	1.93e6	176.68	**0.00**	**0.00**	**0.00**	7.23	83.71	2.11e3	1.39e8	4.22e5
	Best	**10.83**	1.13e5	1.39e5	13.03	2.31e3	3.26e4	**58.67**	320.01	200.04	208.41	**100.30**	413.95	1.08e3	1.20e4	**8.29e3**
	Worst	12.31	**1.11e6**	**8.68e5**	25.47	1.50e4	3.72e5	**254.35**	325.65	200.75	213.70	100.53	678.17	**1.21e3**	5.61e4	3.78e4
$ImpSCA_4$	*Mean*	11.84	**4.14e5**	**4.00e5**	**16.57**	5.80e3	9.00e4	149.05	322.04	200.20	210.83	100.45	456.55	**1.15e3**	**2.67e4**	2.00e4
	Median	11.88	**3.65e5**	**3.18e5**	**16.04**	**5.20e3**	8.25e4	155.30	321.94	200.17	210.81	100.45	425.20	1.16e3	**2.61e4**	**1.80e4**
	Std	0.34	**2.16e5**	**2.00e5**	**2.32**	**2.37e3**	5.70e4	**59.33**	1.27	0.11	1.17	0.05	62.77	**29.13**	8.91e3	6.37e3

5.1.5. Computational complexity analysis

In this subsection, we realized computational complexity analysis of the proposed ImpSCAs. The computational complexity is based on initialization, updating of search agents, and function evaluations. The time complexity analysis for the proposed ImpSCA versions is defined as follows.

The computational complexity in the initialization phase is equal to $O(N)$ due to the fact that there are N search agents in the population. In the main loop of the ImpSCAs, the computational complexity in the update mechanism is equal to $O(T \times N \times D)$ based on the updating of the positions of search agents. Also, the time complexity is equal to $O(T \times N)$ for finding the best solution in the population. Here, N denotes the number of search agents, T is the number of maximum iteration, and D stands for the problem dimension. As a result, the computational complexity of the versions of ImpSCA algorithm is found as $O(N \times (1 + T + TD))$.

5.2. Results of CEC2014 benchmark problems using ImpSCAs

5.2.1. Comparison with original SCA

In this section, we have utilized well-known numerical optimization problems including CEC 2014 test suite to evaluate the performances of the proposed ImpSCAs. This test suite consists of thirty benchmark functions which are minimization problems. These benchmark functions are categorized into four groups: unimodal (FN1–FN3), simple multimodal (FN4–FN16), hybrid (FN17–FN22), and composition functions (FN23–FN30). Unlike benchmark functions, real-world optimization problems have different difficulties such as constraints and discrete-valued decision variables. While optimizing the discrete-valued search fields, each candidate solution is rounded to the nearest integer. In case of constraint violation of each candidate solution, the best candidate solution is selected from the sorted list of candidate solutions according to increasing constraint violation.

Table 4
50D CEC2014 benchmark results for SCA and ImpSCA versions.

		FN1	FN2	FN3	FN4	FN5	FN6	FN7	FN8	FN9	FN10	FN11	FN12	FN13	FN14	FN15
SCA	*Best*	2.82e8	3.77e10	**5.89e4**	4.28e3	21.03	56.70	330.17	418.12	498.46	1.02e4	1.24e4	2.15	3.72	84.53	2.99e4
	Worst	1.02e9	6.52e10	**1.03e5**	9.19e3	21.19	73.10	605.16	550.21	625.46	1.32e4	1.42e4	3.75	5.17	173.24	2.52e5
	Mean	6.34e8	4.94e10	**7.99e4**	6.84e3	21.13	65.32	445.14	489.72	548.25	1.21e4	1.34e4	3.32	4.43	113.44	8.77e4
	Median	6.02e8	4.89e10	**7.94e4**	6.77e3	21.14	65.50	441.88	493.35	548.16	1.21e4	1.35e4	3.37	4.41	113.63	8.16e4
	Std	1.63e8	5.35e9	9.84e3	1.31e3	0.04	3.25	64.12	28.08	27.97	718.36	421.40	0.28	0.29	14.60	3.77e4
$ImpSCA_1$	*Best*	**3.24e7**	**1.26e9**	6.34e4	**288.17**	**20.96**	33.49	**12.78**	221.17	258.05	7.44e3	9.29e3	**1.31**	**0.38**	0.31	**43.98**
	Worst	8.50e7	2.28e9	1.16e5	488.23	21.18	48.01	21.88	**284.84**	334.71	1.00e4	1.12e4	3.00	0.68	**0.58**	92.96
	Mean	5.93e7	**1.78e9**	8.23e4	**321.10**	21.09	40.65	16.88	252.11	288.38	8.73e3	1.04e4	2.47	0.54	**0.37**	**60.44**
	Median	5.86e7	1.79e9	8.67e4	**373.29**	21.10	40.66	16.86	253.74	289.11	8.81e3	1.04e4	2.54	0.55	**0.36**	**59.32**
	Std	1.21e7	2.41e8	1.16e4	**37.84**	0.04	3.12	1.95	14.83	**14.97**	538.08	416.62	0.34	0.06	**0.05**	**8.93**
$ImpSCA_2$	*Best*	1.47e8	2.04e10	8.44e4	1.95e3	**20.96**	46.60	135.86	321.58	354.22	**7.21e3**	**8.48e3**	1.48	1.97	37.27	3.63e3
	Worst	3.34e8	2.57e10	1.18e5	3.25e3	**21.15**	51.42	191.44	376.85	424.36	**8.85e3**	**1.04e4**	**2.67**	2.94	56.51	1.12e4
	Mean	2.65e8	2.32e10	1.04e5	2.77e3	**21.08**	49.45	172.66	355.87	397.17	**8.19e3**	**9.76e3**	**2.16**	2.59	49.54	7.19e3
	Median	2.69e8	2.35e10	1.05e5	2.79e3	**21.09**	49.49	175.86	358.31	397.52	**8.22e3**	**9.85e3**	**2.20**	2.67	49.94	7.32e3
	Std	3.90e7	1.46e9	**7.62e3**	262.16	0.03	1.13	13.25	**13.08**	15.43	365.23	386.93	**0.24**	0.21	3.79	1.75e3
$ImpSCA_3$	*Best*	1.03e9	9.60e10	9.36e4	1.89e4	21.03	66.28	908.92	552.22	583.57	1.14e4	1.25e4	2.66	6.60	222.77	5.01e5
	Worst	2.38e9	1.25e11	1.35e5	3.10e4	21.19	70.19	1.15e3	630.25	677.51	1.29e4	1.38e4	3.97	7.48	288.35	1.21e6
	Mean	1.86e9	1.13e11	1.20e5	2.59e4	21.13	68.43	1.07e3	602.33	642.87	1.22e4	1.33e4	3.31	6.96	255.39	8.75e5
	Median	1.87e9	1.13e11	1.21e5	2.57e4	21.13	68.65	1.08e3	606.48	645.14	1.23e4	1.33e4	3.33	6.95	256.06	8.69e5
	Std	2.98e8	5.98e9	9.47e3	2.17e3	0.03	**1.09**	53.95	16.98	17.67	**357.35**	**344.09**	0.29	0.20	13.88	1.50e5
$ImpSCA_4$	*Best*	3.58e7	1.40e9	6.59e4	300.63	21.04	**32.66**	13.37	**201.32**	**233.71**	7.84e3	9.07e3	1.82	**0.38**	**0.28**	47.20
	Worst	**7.60e7**	**2.20e9**	1.07e5	**473.98**	21.17	**46.64**	**19.56**	290.81	**323.04**	1.00e4	1.09e4	2.94	**0.62**	0.83	**88.89**
	Mean	**5.74e7**	**1.78e9**	8.58e4	379.44	21.11	**39.83**	**16.43**	**249.42**	**286.18**	8.85e3	1.02e4	2.46	**0.52**	0.38	60.98
	Median	**5.65e7**	**1.78e9**	8.45e4	373.48	21.11	**40.11**	**16.47**	**252.48**	**286.03**	8.90e3	1.01e4	2.49	**0.53**	**0.36**	59.90
	Std	**1.03e7**	**1.85e8**	8.94e3	38.19	**0.02**	3.38	**1.71**	18.79	20.49	497.22	444.07	**0.24**	**0.05**	0.09	9.42
		FN16	FN17	FN18	FN19	FN20	FN21	FN22	FN23	FN24	FN25	FN26	FN27	FN28	FN29	FN30
SCA	*Best*	21.76	1.41e7	5.02e8	181.77	1.14e4	4.86e6	1.70e3	576.07	203.98	205.28	103.88	1.85e3	4.17e3	8.22e7	6.69e5
	Worst	22.84	7.69e7	2.16e9	389.89	3.81e4	2.06e7	2.86e3	819.22	362.22	308.97	105.47	2.22e3	7.06e3	2.76e8	3.21e6
	Mean	22.38	4.08e7	1.26e9	270.95	2.33e4	9.33e6	2.28e3	671.51	283.23	267.97	104.70	2.07e3	5.33e3	1.99e8	1.62e6
	Median	22.37	4.17e7	1.27e9	236.68	2.30e4	8.35e6	2.29e3	667.80	277.67	266.02	104.70	2.07e3	5.14e3	2.00e8	1.62e6
	Std	0.26	1.37e7	3.45e8	50.31	6.05e3	3.76e6	238.05	56.47	53.44	19.05	0.38	78.48	695.13	3.96e7	5.25e5
$ImpSCA_1$	*Best*	20.49	7.45e5	2.92e6	**34.92**	7.76e3	**2.70e5**	379.47	373.01	200.17	200.00	**100.36**	**994.80**	2.31e3	1.97e5	5.68e4
	Worst	22.05	**4.08e6**	**1.17e7**	83.08	**2.99e4**	**1.95e6**	**928.64**	388.83	271.61	236.28	100.59	1.37e3	3.36e3	7.21e5	**1.25e5**
	Mean	21.49	2.19e6	6.60e6	50.37	**1.69e4**	9.51e5	**687.51**	380.91	211.75	224.02	100.50	**1.22e3**	2.57e3	**3.94e5**	8.80e4
	Median	21.52	2.11e6	6.76e6	47.42	**1.73e4**	9.07e5	**714.15**	380.98	201.53	228.54	100.50	**1.23e3**	2.53e3	**3.75e5**	8.69e4
	Std	0.32	**7.31e5**	2.11e6	11.64	**4.67e3**	4.35e5	139.93	3.30	20.81	11.53	**0.04**	73.96	188.78	1.20e5	**1.48e4**
$ImpSCA_2$	*Best*	20.88	2.56e6	6.66e7	160.66	2.02e4	8.93e5	835.15	361.46	200.41	**200.00**	105.20	1.06e3	7.69e3	6.80e6	3.95e5
	Worst	**21.88**	1.86e7	3.95e8	229.35	4.36e4	2.95e6	1.36e3	479.74	200.61	**200.00**	200.00	1.74e3	1.00e4	9.71e7	9.38e5
	Mean	**21.46**	9.22e6	2.28e8	195.24	3.35e4	1.69e6	1.18e3	429.81	200.50	**200.00**	148.93	1.49e3	8.96e3	4.62e7	6.99e5
	Median	**21.48**	9.07e6	2.20e8	195.36	3.43e4	1.66e6	1.18e3	430.56	200.51	**200.00**	128.06	1.55e3	8.95e3	4.58e7	6.97e5
	Std	0.23	3.16e6	8.89e7	15.61	5.18e3	4.73e5	**110.31**	22.42	0.04	**0**	40.28	184.06	531.81	1.86e7	1.16e5
$ImpSCA_3$	*Best*	21.71	9.75e7	3.93e9	449.13	2.72e4	1.32e7	3.76e3	**200.00**	**200.00**	**200.00**	130.05	2.26e3	**200.00**	**200.00**	**200.00**
	Worst	22.67	2.62e8	9.53e9	939.52	9.33e4	3.33e7	1.31e4	**200.00**	**200.00**	**200.00**	200.00	2.67e3	1.47e4	1.22e9	2.36e7
	Mean	22.33	1.81e8	7.05e9	745.83	6.12e4	2.17e7	6.46e3	**200.00**	**200.00**	**200.00**	165.03	2.52e3	1.24e4	2.40e7	1.29e7
	Median	22.35	1.91e8	7.05e9	752.47	6.06e4	2.15e7	6.18e3	**200.00**	**200.00**	**200.00**	145.50	2.54e3	1.35e4	2.17e7	1.38e7
	Std	**0.18**	4.16e7	1.35e9	113.02	1.53e4	4.64e6	2.00e3	**0.00**	**0.00**	**0.00**	40.25	84.87	3.67e3	1.71e8	6.73e6
$ImpSCA_4$	*Best*	**20.14**	**6.62e5**	**2.75e6**	36.39	**5.32e3**	2.80e5	**336.92**	371.43	200.15	200.00	100.40	1.08e3	2.21e3	2.08e5	4.79e4
	Worst	21.95	4.46e6	1.24e7	**74.36**	3.23e4	2.35e6	1.02e3	388.80	271.50	234.70	**100.56**	**1.34e3**	**2.90e3**	**6.38e5**	1.30e5
	Mean	21.54	**2.09e6**	**6.53e6**	**48.32**	1.81e4	**8.87e5**	721.18	379.90	217.71	226.14	**100.48**	1.23e3	**2.50e3**	3.97e5	**8.32e4**
	Median	21.62	**2.06e6**	**6.05e6**	**46.40**	1.80e4	**7.75e5**	723.76	379.58	201.57	229.02	**100.48**	1.24e3	**2.51e3**	3.96e5	**8.41e4**
	Std	0.31	7.33e5	**2.08e6**	**9.80**	5.85e3	**4.15e5**	138.63	3.60	26.52	9.92	**0.04**	**60.00**	**117.03**	**1.11e5**	1.55e4

The original SCA and the proposed ImpSCAs have been run 51 times for all benchmark functions at different problem dimension sizes (10D, 30D, and 50D). In the benchmark tests, the population size is used as 10 times the number of the dimension, and the maximum iteration number is 1000. The termination criterion is to reach the maximum number of iterations. The codes of SCA and ImpSCAs have been run on PC with Intel(R) Core(TM) i7-6500U CPU@2.50 GHz/8.00 GB RAM. According to the conditions of CEC 2014, 14 error values were recorded for each function for each run. In the comparison results, five metrics were evaluated such as mean, worst, best, median, and standard deviation. Tables 2–4 summarize the results of the CEC 2014 test suite for 10D, 30D, and 50D, obtained by SCA and ImpSCAs. The best results among the metrics have been emphasized in boldface.

As can be seen from the 10D benchmark results in Table 2, the original SCA has the worst performance according to all metrics. It is clear from the comparison results that $ImpSCA_1$ and $ImpSCA_4$ outperform the original SCA and rest versions of ImpSCA for all benchmarks. It is observed from the 30D benchmark test results in Table 3 that the proposed ImpSCAs show better performance than the original SCA for all functions. In the results of 50D benchmark tests, the worst performance belongs to the original SCA and these results support the better performances of the proposed versions of ImpSCA in terms of all metrics. To show the performance summaries in all metrics and all dimension sizes, we have given the achievement scores of SCA and ImpSCA versions in Table 5. It is clearly evident from these results that the best algorithm is $ImpSCA_1$ in terms of four metrics except standard deviation. $ImpSCA_4$ version has the second-best performance according to the results in the score table.

To compare benchmark results of the proposed ImpSCA versions and original SCA, in terms of statistical significance, Wilcoxon rank-sum test is utilized that is one of the nonparametric tests. This test is a statistical method often used to compare two samples or repeated measurements on a single sample. In Tables 6–8, the p-value scores obtained by Wilcoxon rank-sum test with 5% accuracy from a pair of

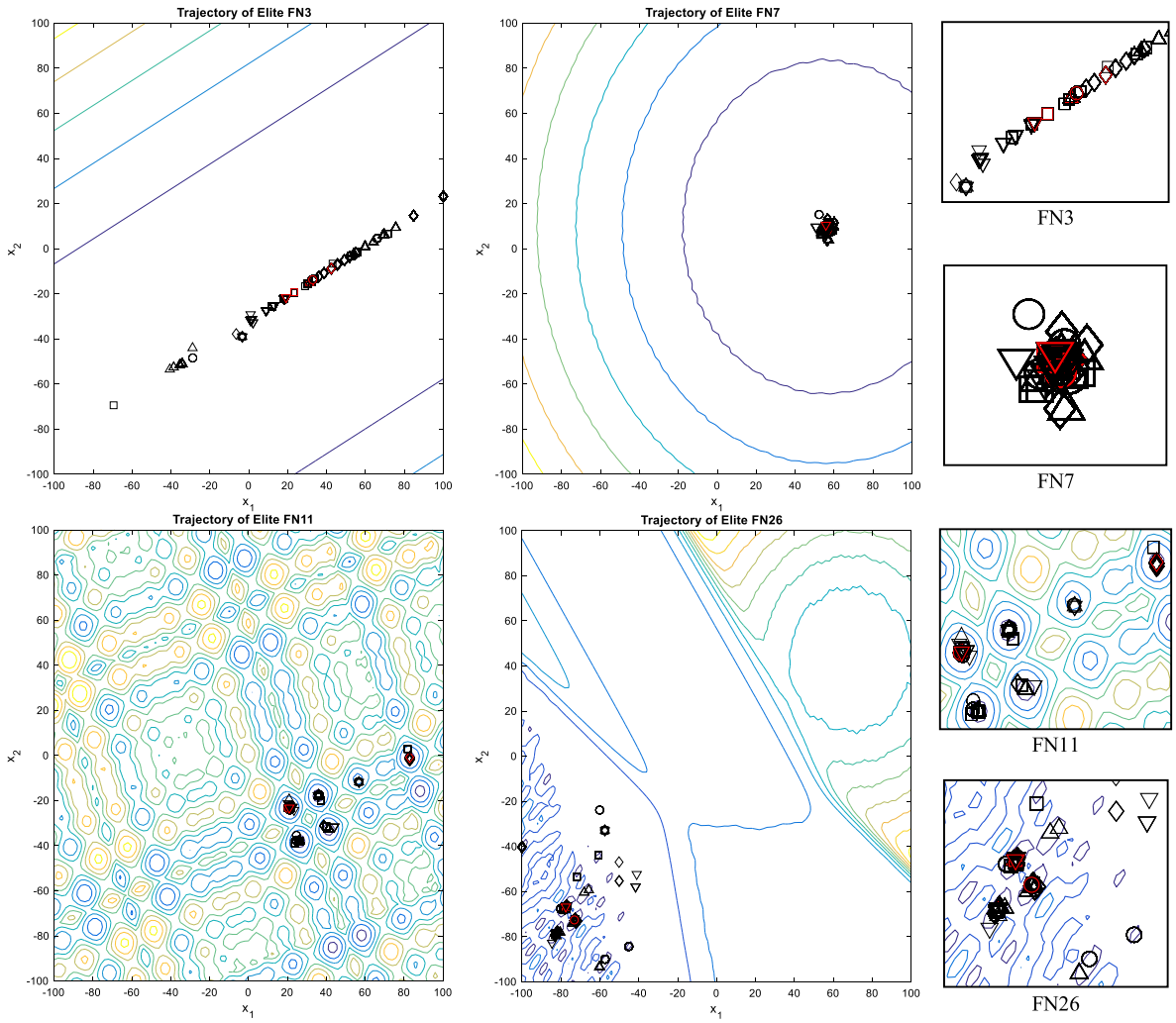


Fig. 5. Trajectory of elite analysis results for selected benchmark functions (contour plots). (◊ : SCA, Δ : ImpSCA$_1$, □ : ImpSCA$_2$, ○ : ImpSCA$_3$, ∇ : ImpSCA$_4$). . (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 5
Score table of the benchmark results for SCA and ImpSCAs.

	SCA	ImpSCA$_1$	ImpSCA$_2$	ImpSCA$_3$	ImpSCA$_4$
Best	1	**40**	11	12	32
Worst	1	**35**	18	7	33
Mean	1	**40**	16	7	33
Median	1	**38**	13	9	36
Std	2	28	16	18	**35**
Total	6	**181**	74	53	169

samples for two algorithms of 50 independent runs to test the null hypothesis for benchmark functions are summarized for different problem dimension sizes (10D, 30D, and 50D). Here, we used a confidence level of 0.95 for statistical analysis, and p-values more than or equal to 0.05 are shown in boldface. Looking at the results in these tables, p-values show that there are significant differences between the results obtained by the SCA and the proposed ImpSCA$_1$, ImpSCA$_2$, and ImpSCA$_4$ for all benchmark problems. However, there is no significant difference between ImpSCA$_3$ and SCA for only 4 benchmarks in the 10D *p*-value table results, only 5 benchmarks in the 30D *p*-value table results, and only 5 benchmarks in the 50D *p*-value table results.

5.2.2. Comparison with other algorithms

In this subsection, the performances of the proposed ImpSCAs are compared with that of the state-of-the-art improved SCA algorithms and other population-based metaheuristic optimization algorithms. We used Harris Hawks Optimizer (HHO) (Heidari et al., 2019), Grey Wolf Optimization (GWO) (Mirjalili et al., 2014), and Moth–Flame Optimization (MFO) (Mirjalili, 2015b) algorithms in the comparison with population-based metaheuristics. For the comparison work with the other improved SCA algorithms in the literature, modified Sine Cosine Algorithm (m-SCA) (Nayak et al., 2018), Opposition based Sine Cosine Algorithm (OBSCA) (Abd Elaziz et al., 2017), and Memory Guided Sine Cosine Algorithm (MG-SCA) (Gupta et al., 2020) were selected. The parameters of these algorithms were set as in the original papers. Tables 9–10 summarize the results for the 30D CEC 2014 benchmarks.

The results presented in these tables are ranked according to the error values of benchmark functions. At the same time, the average and overall ranks of all algorithms are given at the end of the tables. Looking at the average and overall ranks in Table 9, the proposed ImpSCA$_1$ and ImpSCA$_4$ have outperformed m-SCA, OBSCA, MGSCA taken from the literature. The performance of ImpSCA$_2$ is better than that of m-SCA and OBSCA except for MGSCA. The ranking results in Table 10, which includes comparisons made with HHO, GWO, and MFO, show that the first and second algorithms are the proposed ImpSCA$_1$ and ImpSCA$_4$.

In the light of all these results, the proposed four different SCA versions (ImpSCAs) have a more successful search structure than the original SCA structure for CEC2014 test problems with different dimensions. Again, the results of the comparison with different heuristic

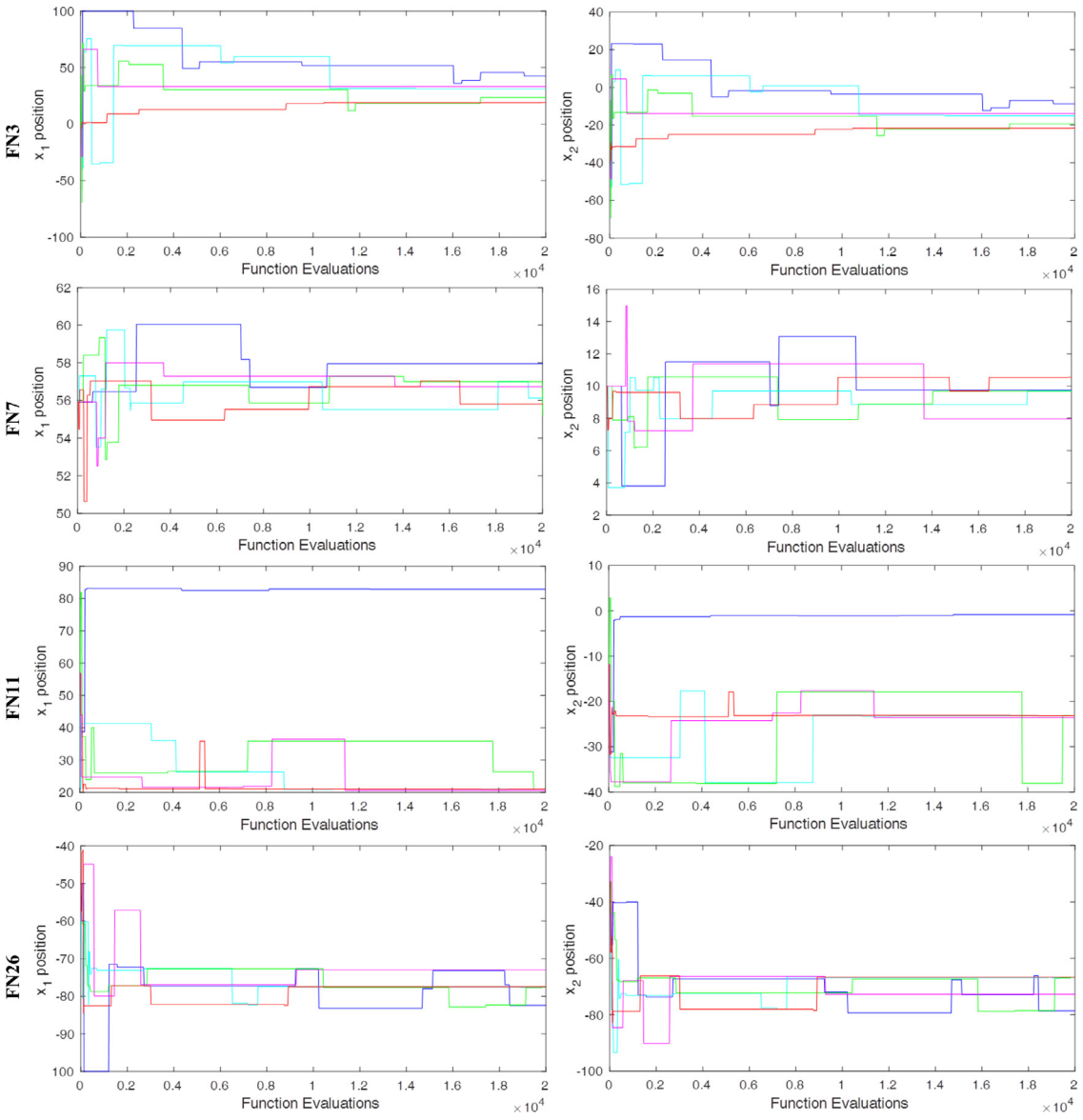


Fig. 6. Trajectory of elite analysis results for benchmark functions (x_1 and x_2 position graphs). (—: SCA, —: ImpSCA$_1$, —: ImpSCA$_2$, —: ImpSCA$_3$, —: ImpSCA$_4$). . (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

algorithms and different modified SCA structures show that the two proposed versions (ImpSCA$_1$ and ImpSCA$_4$) are the two most successful algorithms among all versions of ImpSCA in this study. The update structure proposed in the ImpSCA$_1$ version is based on combining the mechanisms with the Sine and Cosine functions and updating the position of the search agent according to their average. In ImpSCA$_4$ version, two trial vectors found according to Sine and Cosine functions are updated by crossover operation. Both mechanisms used here provide a fast approach to the global solution of the optimization problem, especially in the exploration phase, and a more effective search in the exploitation phase. In ImpSCA$_2$ version, unlike ImpSCA$_1$, a random search agent's position in the population was used instead of the global best in the update mechanism. The third of ImpSCA versions is based on the weighted average of the updated positions found by mechanisms with the Sine and Cosine functions. The mutation mechanism common to all ImpSCA versions allows to avoid local optimal solutions, the new boundary checking mechanism provides more effective alternative solutions especially in case of exceeding the limit values in the exploration phase, and the greedy selection mechanism enables the search agents in the population to approach the global point more effectively in the exploration phase.

In our opinion, the success achieved in the first proposed ImpSCA version (ImpSCA$_1$) is due to the simultaneous use of the effects of Sine and Cosine mathematical functions in the new update mechanism. In the same way, the good results obtained by the last version of ImpSCA (ImpSCA$_4$) are also due to the crossover mechanism proposed in the improved update mechanism. As can be seen from the average distance analysis in the proposed ImpSCA versions, candidate solutions move close to each other due to the added selection mechanism, which reduces diversity. However, the obtained results show us that although the diversity decreases, search agents acting at close range successfully approach the global solution. In cases where the Sine and Cosine functions in the update mechanism have a zero value due to their nature, the search agent takes its previous value and no update.

5.3. Solving feature selection problem using ImpSCAs

5.3.1. Comparison with original SCA

In this study, we have adapted all versions of ImpSCA according to the binary form for feature selection problems from the literature. KNN classifier (Tiwari, 2017; Acuña and Rodriguez, 2004) has been

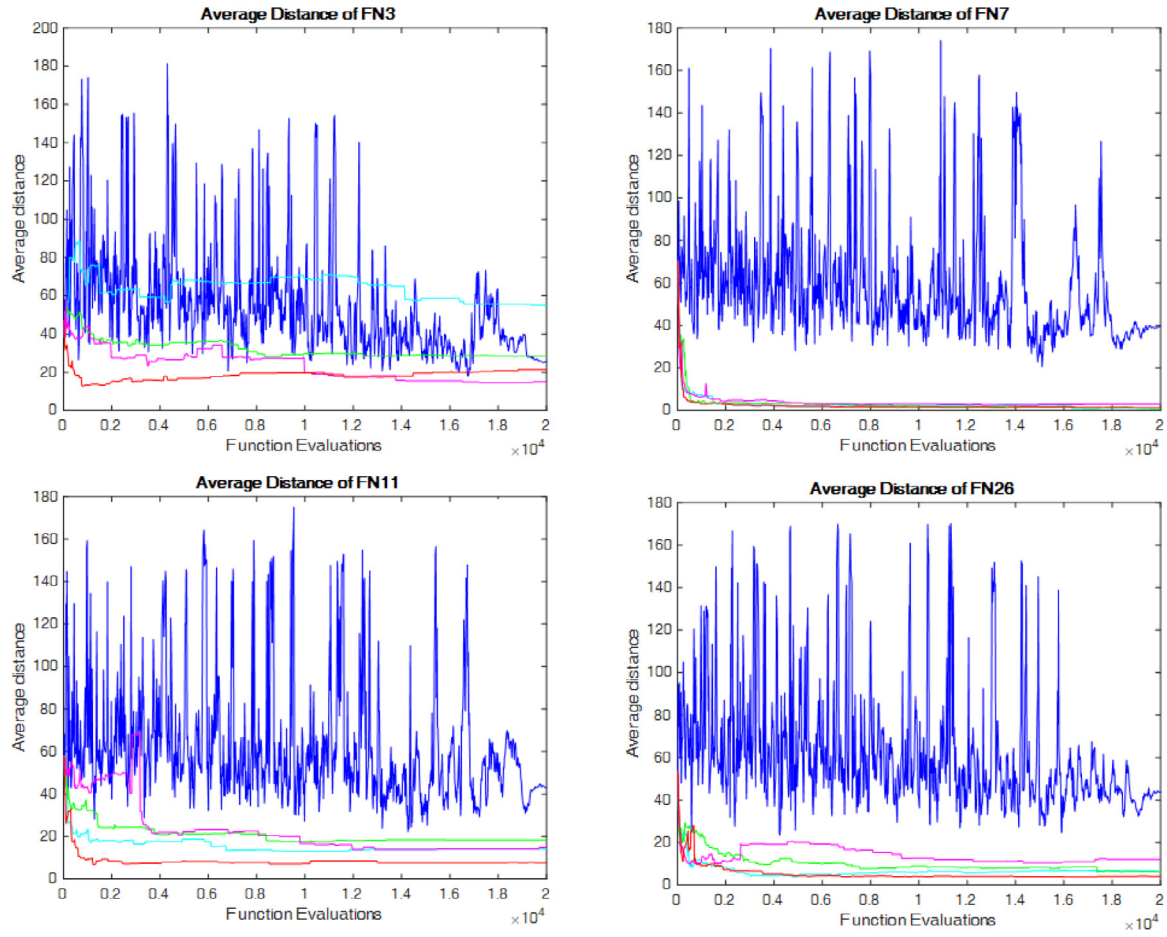


Fig. 7. Average distance analysis between search agents. (—: SCA, —: ImpSCA$_1$, —: ImpSCA$_2$, —: ImpSCA$_3$, —: ImpSCA$_4$). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 6
P-values of the Wilcoxon rank-sum test with 5% significance for 10D CEC2014.

No	ImpSCA$_1$ versus SCA	ImpSCA$_2$ versus SCA	ImpSCA$_3$ versus SCA	ImpSCA$_4$ versus SCA
FN1	2.95E−07	2.95E−07	1.60E−17	3.30E−18
FN2	3.30E−18	3.30E−18	3.94E−18	3.30E−18
FN3	9.55E−05	9.55E−05	5.65E−14	2.03E−03
FN4	7.96E−18	7.96E−18	3.30E−18	4.70E−18
FN5	4.74E−10	4.74E−10	**6.76E−01**	3.52E−06
FN6	7.51E−18	7.51E−18	4.60E−14	3.30E−18
FN7	3.30E−18	3.30E−18	3.30E−18	3.30E−18
FN8	5.95E−18	5.95E−18	1.20E−17	3.30E−18
FN9	3.01E−17	3.01E−17	3.38E−14	3.30E−18
FN10	3.30E−18	3.30E−18	2.49E−03	3.30E−18
FN11	4.43E−18	4.43E−18	**1.37E−01**	5.29E−18
FN12	3.50E−18	3.50E−18	**7.89E−01**	3.19E−17
FN13	4.70E−18	4.70E−18	6.30E−18	3.30E−18
FN14	3.30E−18	3.30E−18	3.30E−18	3.30E−18
FN15	3.50E−18	3.50E−18	8.44E−18	3.30E−18
FN16	1.12E−12	1.12E−12	4.55E−09	5.61E−18
FN17	7.51E−18	7.51E−18	2.70E−15	4.70E−18
FN18	1.69E−17	1.69E−17	**1.83E−01**	1.90E−17
FN19	3.94E−18	3.94E−18	1.11E−16	3.30E−18
FN20	1.88E−02	1.88E−02	2.57E−13	2.12E−08
FN21	9.55E−12	9.55E−12	1.82E−06	4.24E−16
FN22	2.00E−11	2.00E−11	2.56E−15	3.30E−18
FN23	5.19E−20	5.19E−20	1.39E−20	3.30E−18
FN24	2.30E−16	2.30E−16	1.07E−17	3.30E−18
FN25	2.65E−10	2.65E−10	2.02E−03	1.75E−15
FN26	3.30E−18	3.30E−18	3.30E−18	3.29E−18
FN27	4.70E−18	4.70E−18	5.36E−07	3.59E−16
FN28	8.86E−11	8.86E−11	6.31E−17	1.45E−09
FN29	5.61E−18	5.61E−18	4.18E−18	3.72E−18
FN30	1.16E−10	1.16E−10	3.30E−18	2.54E−10

Table 7
P-values of the Wilcoxon rank-sum test with 5% significance for 30D CEC2014.

No	$ImpSCA_1$ versus SCA	$ImpSCA_2$ versus SCA	$ImpSCA_3$ versus SCA	$ImpSCA_4$ versus SCA
FN1	8.44E−18	8.44E−18	3.30E−18	3.30E−18
FN2	3.30E−18	3.30E−18	3.30E−18	3.30E−18
FN3	1.91E−11	1.91E−11	1.60E−17	4.13E−06
FN4	3.30E−18	3.30E−18	3.30E−18	3.30E−18
FN5	1.60E−07	1.60E−07	**2.74E−01**	1.17E−05
FN6	3.30E−18	3.30E−18	1.28E−02	3.30E−18
FN7	3.30E−18	3.30E−18	3.30E−18	3.30E−18
FN8	3.30E−18	3.30E−18	6.30E−18	3.30E−18
FN9	3.30E−18	3.30E−18	1.14E−15	3.30E−18
FN10	3.30E−18	3.30E−18	2.09E−02	3.30E−18
FN11	3.30E−18	3.30E−18	4.19E−02	3.30E−18
FN12	3.30E−18	3.30E−18	**1.51E−01**	1.42E−17
FN13	3.30E−18	3.30E−18	3.30E−18	3.30E−18
FN14	3.30E−18	3.30E−18	3.30E−18	3.30E−18
FN15	3.30E−18	3.30E−18	3.30E−18	3.30E−18
FN16	5.03E−17	5.03E−17	**5.69E−01**	4.75E−17
FN17	6.68E−18	6.68E−18	7.92E−17	3.30E−18
FN18	3.30E−18	3.30E−18	6.51E−10	3.30E−18
FN19	1.48E−02	1.48E−02	5.61E−18	3.30E−18
FN20	5.44E−12	5.44E−12	8.94E−18	1.26E−14
FN21	3.30E−18	3.30E−18	5.97E−17	3.30E−18
FN22	3.30E−18	3.30E−18	7.96E−18	3.30E−18
FN23	6.31E−17	6.31E−17	1.39E−20	3.30E−18
FN24	3.27E−18	3.27E−18	1.39E−20	8.48E−14
FN25	1.39E−20	1.39E−20	**9.58E−02**	3.30E−18
FN26	2.35E−14	2.35E−14	7.17E−18	3.30E−18
FN27	1.57E−15	1.57E−15	1.04E−04	1.12E−12
FN28	2.87E−16	2.87E−16	2.77E−10	3.30E−18
FN29	3.30E−18	3.30E−18	**3.95E−01**	3.30E−18
FN30	1.24E−12	1.24E−12	3.30E−18	3.30E−18

Table 8
P-values of the Wilcoxon rank-sum test with 5% significance for 50D CEC2014.

No	$ImpSCA_1$ versus SCA	$ImpSCA_2$ versus SCA	$ImpSCA_3$ versus SCA	$ImpSCA_4$ versus SCA
FN1	1.13E−17	1.13E−17	3.30E−18	3.30E−18
FN2	3.30E−18	3.30E−18	3.30E−18	3.30E−18
FN3	1.84E−16	1.84E−16	6.30E−18	3.68E−03
FN4	3.30E−18	3.30E−18	3.30E−18	3.30E−18
FN5	4.93E−09	4.93E−09	**7.05E−01**	4.05E−03
FN6	3.30E−18	3.30E−18	4.84E−10	3.30E−18
FN7	3.30E−18	3.30E−18	3.30E−18	3.30E−18
FN8	3.30E−18	3.30E−18	3.30E−18	3.30E−18
FN9	3.30E−18	3.30E−18	5.95E−18	3.30E−18
FN10	3.30E−18	3.30E−18	**2.75E−01**	3.30E−18
FN11	3.30E−18	3.30E−18	**3.77E−01**	3.30E−18
FN12	2.01E−17	2.01E−17	**7.23E−01**	8.38E−17
FN13	3.30E−18	3.30E−18	3.30E−18	3.30E−18
FN14	3.30E−18	3.30E−18	3.30E−18	3.30E−18
FN15	3.30E−18	3.30E−18	3.30E−18	3.30E−18
FN16	6.30E−18	6.30E−18	**2.95E−01**	1.69E−17
FN17	4.43E−18	4.43E−18	3.30E−18	3.30E−18
FN18	3.30E−18	3.30E−18	3.30E−18	3.30E−18
FN19	9.89E−14	9.89E−14	3.30E−18	3.30E−18
FN20	4.73E−12	4.73E−12	1.01E−17	1.19E−04
FN21	3.30E−18	3.30E−18	5.96E−17	3.30E−18
FN22	3.30E−18	3.30E−18	3.30E−18	3.30E−18
FN23	3.30E−18	3.30E−18	1.39E−20	3.30E−18
FN24	3.30E−18	3.30E−18	1.39E−20	2.75E−11
FN25	1.39E−20	1.39E−20	1.39E−20	2.17E−16
FN26	4.64E−18	4.64E−18	8.66E−08	3.30E−18
FN27	3.30E−18	3.30E−18	3.30E−18	3.30E−18
FN28	3.30E−18	3.30E−18	2.21E−13	3.30E−18
FN29	3.94E−18	3.94E−18	1.36E−18	3.30E−18
FN30	4.48E−16	4.48E−16	1.08E−15	3.30E−18

used to evaluate the selected feature subsets. As the objective function of the feature selection problem, two parameters were determined: the number of selected features and the classification accuracy.

Firstly, a candidate solution is represented by a vector of ones and zeros, where ones stand for that the suitable feature is selected and zeros denote that the suitable feature is not selected. The dimension of the candidate solution vector is the number of features in the dataset. A wrapper model based on KNN classifier for feature selection is used in this work. The fitness function utilized to evaluate the candidate solutions in the proposed ImpSCA versions is given below:

$$F_i = \alpha E\left(D_i\right) + (1-\alpha)\frac{N_{f,i}}{N_f} \tag{16}$$

where F_i indicates fitness value for i_{th} candidate solution, α denotes the parameter in the range [0,1], $E\left(D_i\right)$ stands for the classification error rate, D_i represents the selection decision of the features for i_{th} candidate solution, N_f indicates the total number of features in the dataset, and $N_{f,i}$ denotes the number of selected features for i_{th} candidate solution.

The proposed ImpSCA versions were evaluated on ten feature selection datasets taken from UCI data repository (Lichman, 2013). In Table 11, the characteristics of UCI datasets are given. The selected datasets are from different areas, such as biology, chemistry, physics, and games. In the tests of the proposed ImpSCAs for feature selection, each dataset was randomly divided into two parts: training (80% of the dataset) and testing (20% of the dataset). The parameters were adjusted as summarized in Table 12.

For KNN classifier used in the wrapper feature selection model, K value was selected as 5 according to the trial and error based experiments (Emary et al., 2016a). Table 13 summarizes the best classification accuracy and selected attributes obtained by SCA and ImpSCAs for five runs. The best results have been emphasized in boldface. In terms of the accuracy results, $ImpSCA_4$ has the best performance in four datasets.

The other versions of ImpSCA come next by obtaining the best accuracy results in two datasets. The algorithm which has the worst performance is the original SCA. In terms of the number of selected features, SCA is the best, but the decreasing number of attributes has not improved the accuracy of the KNN classifier. This is due to the fact that after the reduction in the number of features, unsuitable features in the dataset have been selected.

In Table 14, the statistical results regarding fitness values obtained by SCA and ImpSCAs are presented. Here, $ImpSCA_4$ version is first in five datasets in terms of mean fitness results. SCA and $ImpSCA_1$ have the best mean fitness values in only two datasets. According to the best fitness results, $ImpSCA_4$ has the minimum values in four datasets again. The best fitness values in two datasets belong to the remaining versions of ImpSCA. The original SCA is the best in only one dataset. In terms of the worst statistical results, $ImpSCA_4$ and $ImpSCA_1$ versions outperform the other versions of ImpSCA and SCA.

As a result, it is evident from this table that $ImpSCA_4$ version has the best performance according to the statistical results of the fitness values. Table 15 summarizes the statistical results of the accuracy values obtained by SCA and ImpSCAs for five independent runs.

From this table, it can be seen that $ImpSCA_4$ outperforms the original SCA and the other ImpSCA versions in all statistical metrics. In terms of the mean accuracy results, $ImpSCA_4$ is the first in 5 out of 10 datasets. The second-best performance belongs to $ImpSCA_1$ version. According to the best classification accuracy results, $ImpSCA_4$ has the best performance in four datasets. The other versions of ImpSCA have the same performances (2 out of 10 datasets). As the last statistical metric (worst accuracy), $ImpSCA_4$ still outperforms all other algorithms in 5 out of 10 datasets, and it is ranked first with the 2nd and 3rd versions of ImpSCA for SonarEW dataset.

To show the significant differences between independent runs of the original SCA and of the proposed ImpSCAs for feature selection problem dataset, Wilcoxon rank-sum test with 5% significance was used in terms of the fitness values. Table 16 presents the p-values of the Wilcoxon rank-sum test with 5% significance for fitness values obtained by SCA and ImpSCA versions. In this test, we utilized a confidence level of 95% for statistical analysis, and p-values more than or equal to 0.05 are shown in boldface. These results show that there are especially

Table 9
Performance comparison results of m-SCA, OBSCA, MGSCA, and ImpSCA versions for 30D CEC2014 problems.

		m-SCA (Nayak et al., 2018)	OBSCA (Abd Elaziz et al., 2017)	MGSCA (Gupta et al., 2020)	$ImpSCA_1$	$ImpSCA_2$	$ImpSCA_3$	$ImpSCA_4$
FN1	*Mean*	2.12E+08	4.65E+08	2.92E+07	2.09E+07	8.54E+07	6.63E+08	2.30E+07
	Std	5.56E+07	1.22E+08	2.07E+07	7.17E+06	1.62E+07	7.60E+07	5.65E+06
	Rank	5	6	3	1	4	7	2
FN2	*Mean*	1.57E+10	2.20E+10	2.26E+09	2.37E+08	3.92E+09	5.00E+10	2.41E+08
	Std	2.48E+09	4.34E+09	1.69E+09	4.37E+07	6.36E+08	3.59E+09	4.18E+07
	Rank	5	6	3	1	4	7	2
FN3	*Mean*	3.97E+04	5.28E+04	1.77E+04	2.84E+04	4.79E+04	6.10E+04	3.01E+04
	Std	7.56E+03	1.29E+04	6.63E+03	6.77E+03	4.00E+03	3.97E+03	7.27E+03
	Rank	4	6	1	2	5	7	3
FN4	*Mean*	9.86E+02	1.61E+03	2.76E+02	1.77E+02	4.97E+02	4.50E+03	1.78E+02
	Std	3.02E+02	5.26E+02	6.55E+01	1.45E+01	5.09E+01	6.02E+02	1.85E+01
	Rank	5	6	3	1	4	7	2
FN5	*Mean*	2.09E+01	2.09E+01	2.04E+01	2.09E+01	2.09E+01	2.10E+01	2.09E+01
	Std	3.80E−02	5.60E−02	1.44E−01	7.00E−02	7.00E−02	5.00E−02	6.00E−02
	Rank	3	3	1	4	2	5	3
FN6	*Mean*	3.35E+01	3.05E+01	1.94E+01	1.67E+01	2.31E+01	3.55E+01	1.70E+01
	Std	2.58E+00	1.20E+00	2.89E+00	1.83E+00	1.14E+00	1.07E+00	2.03E+00
	Rank	6	5	3	1	4	7	2
FN7	*Mean*	1.15E+02	1.83E+02	1.99E+01	3.75E+00	3.78E+01	4.53E+02	3.89E+00
	Std	1.98E+01	3.19E+01	1.18E+01	5.40E−01	5.06E+00	3.59E+01	7.20E−01
	Rank	5	6	3	1	4	7	2
FN8	*Mean*	2.36E+02	2.53E+02	1.07E+02	8.87E+01	1.50E+02	3.02E+02	9.00E+01
	Std	1.48E+01	1.94E+01	2.14E+01	9.55E+00	9.57E+00	1.17E+01	9.91E+00
	Rank	5	6	3	1	4	7	2
FN9	*Mean*	2.75E+02	3.04E+02	1.39E+02	1.16E+02	1.79E+02	3.11E+02	1.17E+02
	Std	1.61E+01	1.61E+01	2.56E+01	1.20E+01	9.53E+00	1.49E+01	8.98E+00
	Rank	5	6	3	1	4	7	2
FN10	*Mean*	5.99E+03	4.54E+03	2.82E+03	3.40E+03	3.23E+03	6.05E+03	3.45E+03
	Std	4.53E+02	4.87E+02	6.83E+02	4.30E+02	2.03E+02	3.37E+02	4.91E+02
	Rank	6	5	1	3	2	7	4
FN11	*Mean*	7.00E+03	5.96E+03	3.30E+03	4.70E+03	4.28E+03	6.92E+03	4.80E+03
	Std	3.42E+02	4.31E+02	6.26E+02	4.16E+02	3.21E+02	3.12E+02	3.58E+02
	Rank	7	5	1	3	2	6	4
FN12	*Mean*	2.44E+00	2.12E+00	6.30E−01	1.65E+00	1.42E+00	2.51E+00	1.69E+00
	Std	3.50E−01	3.10E−01	3.40E−01	2.40E−01	2.10E−01	2.00E−01	2.30E−01
	Rank	6	5	1	3	2	7	4
FN13	*Mean*	2.94E+00	3.24E+00	5.50E−01	4.00E−01	6.50E−01	5.90E+00	3.90E−01
	Std	3.70E−01	3.30E−01	8.00E−02	5.00E−02	6.00E−02	2.50E−01	4.00E−02
	Rank	5	6	3	2	4	7	1
FN14	*Mean*	4.42E+01	5.23E+01	2.34E+00	2.60E−01	1.66E+01	1.55E+02	2.60E−01
	Std	7.81E+00	1.13E+01	3.31E+00	3.00E−02	2.44E+00	1.25E+01	3.00E−02
	Rank	4	5	2	1	3	6	1
FN15	*Mean*	1.92E+03	2.12E+04	8.72E+01	1.82E+01	9.23E+01	5.52E+04	1.80E+01
	Std	1.43E+03	1.11E+04	1.01E+02	1.50E+00	1.60E+01	1.58E+04	1.49E+00
	Rank	5	6	3	2	4	7	1
FN16	*Mean*	1.28E+01	1.27E+01	1.16E+01	1.18E+01	1.20E+01	1.28E+01	1.18E+01
	Std	2.20E−01	3.50E−01	6.90E−01	2.80E−01	2.00E−01	1.50E−01	3.40E−01
	Rank	6	5	1	2	4	7	3
FN17	*Mean*	5.37E+06	1.31E+07	9.56E+05	4.29E+05	8.91E+05	2.39E+07	4.14E+05
	Std	2.76E+06	5.63E+06	7.62E+05	2.41E+05	3.57E+05	7.99E+06	2.16E+05
	Rank	5	6	4	2	3	7	1
FN18	*Mean*	1.43E+08	2.36E+08	1.48E+05	4.43E+05	5.45E+06	3.16E+08	4.00E+05
	Std	8.38E+07	1.19E+08	9.00E+05	2.27E+05	3.70E+06	9.22E+07	2.00E+05
	Rank	5	6	1	3	4	7	2
FN19	*Mean*	9.42E+01	1.27E+02	2.28E+01	1.66E+01	7.89E+01	1.74E+02	1.66E+01
	Std	2.63E+01	4.42E+01	1.43E+01	2.37E+00	1.30E+01	2.27E+01	2.32E+00
	Rank	5	6	3	2	4	7	1
FN20	*Mean*	9.51E+03	2.14E+04	4.24E+03	5.61E+03	2.19E+04	5.00E+04	5.80E+03
	Std	3.62E+03	7.96E+03	3.82E+03	2.47E+03	4.59E+03	1.53E+04	2.37E+03
	Rank	4	5	1	2	6	7	3
FN21	*Mean*	1.48E+06	2.14E+06	2.35E+05	8.29E+04	1.57E+05	5.56E+06	9.00E+04
	Std	6.95E+05	1.45E+06	2.39E+05	4.30E+04	4.79E+04	1.93E+06	5.70E+04
	Rank	5	6	4	1	3	7	2

(*continued on next page*)

Table 9 (*continued*).

		m-SCA (Nayak et al., 2018)	OBSCA (Abd Elaziz et al., 2017)	MGSCA (Gupta et al., 2020)	ImpSCA$_1$	ImpSCA$_2$	ImpSCA$_3$	ImpSCA$_4$
	Mean	7.54E+02	8.91E+02	3.39E+02	1.40E+02	3.15E+02	1.40E+03	1.49E+02
FN22	*Std*	1.46E+02	1.69E+02	1.78E+02	6.14E+01	5.14E+01	1.77E+02	5.93E+01
	Rank	5	6	4	1	3	7	2
	Mean	3.66E+02	3.82E+02	3.29E+02	3.22E+02	3.47E+02	2.00E+02	3.22E+02
FN23	*Std*	1.16E+01	2.70E+01	4.03E+00	7.90E−01	5.21E+00	0.00E+00	1.27E+00
	Rank	6	7	4	2	5	1	3
	Mean	2.00E+02	2.00E+02	2.00E+02	2.00E+02	2.00E+02	2.00E+02	2.00E+02
FN24	*Std*	6.90E−02	1.00E−03	1.00E−03	1.00E−01	1.00E−02	0.00E+00	1.10E−01
	Rank	1	1	1	3	2	1	3
	Mean	2.26E+02	2.25E+02	2.11E+02	2.11E+02	2.00E+02	2.00E+02	2.11E+02
FN25	*Std*	8.89E+00	1.43E+01	2.82E+00	1.33E+00	0.00E+00	0.00E+00	1.17E+00
	Rank	5	4	3	2	1	1	2
	Mean	1.02E+02	1.04E+02	1.01E+02	1.00E+02	1.04E+02	1.30E+02	1.00E+02
FN26	*Std*	5.30E−01	5.50E−01	1.50E−01	4.00E−02	7.10E−01	7.23E+00	5.00E−02
	Rank	4	6	3	1	5	7	2
	Mean	8.28E+02	5.07E+02	8.19E+02	4.44E+02	4.71E+02	8.22E+02	4.57E+02
FN27	*Std*	3.39E+02	3.13E+01	9.17E+01	5.35E+01	6.71E+01	8.37E+01	6.28E+01
	Rank	7	4	5	1	3	6	2
	Mean	1.98E+03	1.47E+03	9.68E+02	1.16E+03	2.96E+03	5.30E+03	1.15E+03
FN28	*Std*	2.96E+02	7.60E+01	1.06E+02	5.01E+01	3.60E+02	2.11E+03	2.91E+01
	Rank	5	4	1	3	6	7	2
	Mean	1.04E+07	9.45E+06	1.19E+06	2.79E+04	3.53E+05	1.36E+08	2.67E+04
FN29	*Std*	5.39E+06	4.42E+06	3.25E+06	6.59E+03	2.09E+05	1.39E+08	8.91E+03
	Rank	6	5	4	2	3	7	1
	Mean	2.38E+05	3.17E+05	1.92E+04	1.93E+04	1.25E+05	1.53E+06	2.00E+04
FN30	*Std*	1.01E+05	9.86E+04	8.25E+03	4.99E+03	4.23E+04	4.22E+05	6.37E+03
	Rank	5	6	1	2	4	7	3
Average rank		5.00	5.30	2.47	1.87	3.60	6.23	2.23
Overall rank		5	6	3	1	4	7	2

Table 10
Performance comparison results of GWO, MFO, HHO and ImpSCA versions for 30D CEC2014 problems.

		GWO (Mirjalili et al., 2014)	MFO (Mirjalili, 2015b)	HHO (Heidari et al., 2019)	ImpSCA$_1$	ImpSCA$_2$	ImpSCA$_3$	ImpSCA$_4$
	Mean	6.90E+07	7.59E+07	7.22E+07	2.09E+07	8.54E+07	6.63E+08	2.30E+07
FN1	*Std*	5.38E+07	9.77E+07	3.02E+07	7.17E+06	1.62E+07	7.60E+07	5.65E+06
	Rank	3	5	4	1	6	7	2
	Mean	2.18E+09	1.36E+10	1.80E+08	2.37E+08	3.92E+09	5.00E+10	2.41E+08
FN2	*Std*	2.47E+09	8.42E+09	5.41E+07	4.37E+07	6.36E+08	3.59E+09	4.18E+07
	Rank	4	6	1	2	5	7	3
	Mean	3.12E+04	8.99E+04	4.16E+04	2.84E+04	4.79E+04	6.10E+04	3.01E+04
FN3	*Std*	8.60E+03	4.98E+04	1.07E+04	6.77E+03	4.00E+03	3.97E+03	7.27E+03
	Rank	3	7	4	1	5	6	2
	Mean	2.62E+02	1.14E+03	2.95E+02	1.77E+02	4.97E+02	4.50E+03	1.78E+02
FN4	*Std*	8.96E+01	1.13E+03	7.75E+01	1.45E+01	5.09E+01	6.02E+02	1.85E+01
	Rank	3	5	4	1	5	6	2
	Mean	2.09E+01	2.04E+01	2.06E+01	2.09E+01	2.09E+01	2.10E+01	2.09E+01
FN5	*Std*	4.80E−02	1.75E−01	1.29E−01	7.00E−02	7.00E−02	5.00E−02	6.00E−02
	Rank	4	1	2	5	3	6	4
	Mean	1.39E+01	2.40E+01	3.52E+01	1.67E+01	2.31E+01	3.55E+01	1.70E+01
FN6	*Std*	2.24E+00	3.33E+00	2.95E+00	1.83E+00	1.14E+00	1.07E+00	2.03E+00
	Rank	1	5	6	2	4	7	3
	Mean	1.84E+01	1.17E+02	2.74E+00	3.75E+00	3.78E+01	4.53E+02	3.89E+00
FN7	*Std*	1.44E+01	6.91E+01	6.03E−01	5.40E−01	5.06E+00	3.59E+01	7.20E−01
	Rank	4	6	1	2	5	7	3
	Mean	7.77E+01	1.43E+02	1.46E+02	8.87E+01	1.50E+02	3.02E+02	9.00E+01
FN8	*Std*	1.86E+01	3.81E+01	1.71E+01	9.55E+00	9.57E+00	1.17E+01	9.91E+00
	Rank	1	4	5	2	6	7	3
	Mean	9.62E+01	2.23E+02	1.97E+02	1.16E+02	1.79E+02	3.11E+02	1.17E+02
FN9	*Std*	2.59E+01	6.06E+01	2.06E+01	1.20E+01	9.53E+00	1.49E+01	8.98E+00
	Rank	1	6	5	2	4	7	3
	Mean	2.13E+03	3.47E+03	3.86E+03	3.40E+03	3.23E+03	6.05E+03	3.45E+03
FN10	*Std*	6.62E+02	8.85E+02	7.75E+02	4.30E+02	2.03E+02	3.37E+02	4.91E+02
	Rank	1	5	6	3	2	7	4
	Mean	2.81E+03	4.15E+03	5.05E+03	4.70E+03	4.28E+03	6.92E+03	4.80E+03
FN11	*Std*	9.41E+02	6.90E+02	6.97E+02	4.16E+02	3.21E+02	3.12E+02	3.58E+02

(*continued on next page*)

Table 10 (*continued*).

		GWO (Mirjalili et al., 2014)	MFO (Mirjalili, 2015b)	HHO (Heidari et al., 2019)	$ImpSCA_1$	$ImpSCA_2$	$ImpSCA_3$	$ImpSCA_4$
	Rank	1	2	6	4	3	7	5
	Mean	1.52E+00	4.30E−01	1.79E+00	1.65E+00	1.42E+00	2.51E+00	1.69E+00
FN12	*Std*	1.19E+00	2.60E−01	4.22E−01	2.40E−01	2.10E−01	2.00E−01	2.30E−01
	Rank	3	1	6	4	2	7	5
	Mean	4.40E−01	2.21E+00	5.80E−01	4.00E−01	6.50E−01	5.90E+00	3.90E−01
FN13	*Std*	2.90E−01	1.34E+00	1.21E−01	5.00E−02	6.00E−02	2.50E−01	4.00E−02
	Rank	3	6	4	2	5	7	1
	Mean	4.54E+00	3.54E+01	2.99E−01	2.60E−01	1.66E+01	1.55E+02	2.60E−01
FN14	*Std*	8.41E+00	2.47E+01	1.13E−01	3.00E−02	2.44E+00	1.25E+01	3.00E−02
	Rank	3	5	2	1	4	6	1
	Mean	1.25E+02	2.23E+05	6.74E+01	1.82E+01	9.23E+01	5.52E+04	1.80E+01
FN15	*Std*	3.19E+02	5.77E+05	1.59E+01	1.50E+00	1.60E+01	1.58E+04	1.49E+00
	Rank	5	7	3	2	4	6	1
	Mean	1.09E+01	1.27E+01	1.23E+01	1.18E+01	1.20E+01	1.28E+01	1.18E+01
FN16	*Std*	5.80E−01	5.30E−01	3.79E−01	2.80E−01	2.00E−01	1.50E−01	3.40E−01
	Rank	1	6	5	2	4	7	3
	Mean	1.40E+06	3.39E+06	8.41E+06	4.29E+05	8.91E+05	2.39E+07	4.14E+05
FN17	*Std*	1.56E+06	4.07E+06	4.18E+06	2.41E+05	3.57E+05	7.99E+06	2.16E+05
	Rank	4	5	6	2	3	7	1
	Mean	9.21E+06	5.19E+06	4.49E+05	4.43E+05	5.45E+06	3.16E+08	4.00E+05
FN18	*Std*	1.98E+07	3.61E+07	5.61E+05	2.27E+05	3.70E+06	9.22E+07	2.00E+05
	Rank	6	4	3	2	5	7	1
	Mean	4.34E+01	7.36E+01	7.40E+01	1.66E+01	7.89E+01	1.74E+02	1.66E+01
FN19	*Std*	2.62E+01	5.32E+01	3.42E+01	2.37E+00	1.30E+01	2.27E+01	2.32E+00
	Rank	3	4	5	2	6	7	1
	Mean	1.34E+04	5.67E+04	4.03E+04	5.61E+03	2.19E+04	5.00E+04	5.80E+03
FN20	*Std*	9.03E+03	4.34E+04	2.26E+04	2.47E+03	4.59E+03	1.53E+04	2.37E+03
	Rank	3	7	5	1	4	6	2
	Mean	7.16E+05	7.83E+05	2.23E+06	8.29E+04	1.57E+05	5.56E+06	9.00E+04
FN21	*Std*	1.26E+06	1.18E+06	2.20E+06	4.30E+04	4.79E+04	1.93E+06	5.70E+04
	Rank	4	5	6	1	3	7	2
	Mean	3.78E+04	8.67E+04	8.97E+02	1.40E+02	3.15E+02	1.40E+03	1.49E+02
FN22	*Std*	1.65E+04	2.29E+04	2.65E+02	6.14E+01	5.14E+01	1.77E+02	5.93E+01
	Rank	6	7	4	1	3	5	2
	Mean	3.35E+02	3.71E+02	2.00E+02	3.22E+02	3.47E+02	2.00E+02	3.22E+02
FN23	*Std*	1.04E+01	3.98E+01	5.39E−07	7.90E−01	5.21E+00	0.00E+00	1.27E+00
	Rank	4	6	1	2	5	1	3
	Mean	2.00E+02	2.76E+02	2.00E+02	2.00E+02	2.00E+02	2.00E+02	2.00E+02
FN24	*Std*	8.20E−04	2.73E+01	1.62E−03	1.00E−01	1.00E−02	0.00E+00	1.10E−01
	Rank	1	4	1	3	2	1	3
	Mean	2.10E+02	2.14E+02	2.00E+02	2.11E+02	2.00E+02	2.00E+02	2.11E+02
FN25	*Std*	5.34E+00	7.65E+00	6.48E−09	1.33E+00	0.00E+00	0.00E+00	1.17E+00
	Rank	2	4	1	3	1	1	3
	Mean	1.47E+02	1.03E+02	1.16E+02	1.00E+02	1.04E+02	1.30E+02	1.00E+02
FN26	*Std*	5.02E+01	1.50E+00	3.66E+01	4.00E−02	7.10E−01	7.23E+00	5.00E−02
	Rank	7	3	5	1	4	6	2
	Mean	6.16E+02	9.21E+02	2.00E+02	4.44E+02	4.71E+02	8.22E+02	4.57E+02
FN27	*Std*	1.20E+02	2.23E+02	1.95E−07	5.35E+01	6.71E+01	8.37E+01	6.28E+01
	Rank	5	7	1	2	4	6	3
	Mean	1.20E+03	1.12E+03	2.00E+02	1.16E+03	2.96E+03	5.30E+03	1.15E+03
FN28	*Std*	2.62E+02	1.57E+02	1.77E−07	5.01E+01	3.60E+02	2.11E+03	2.91E+01
	Rank	5	2	1	4	6	7	3
	Mean	1.29E+06	3.06E+06	3.14E+05	2.79E+04	3.53E+05	1.36E+08	2.67E+04
FN29	*Std*	4.28E+06	3.62E+06	1.75E+06	6.59E+03	2.09E+05	1.39E+08	8.91E+03
	Rank	5	6	3	2	4	7	1
	Mean	5.20E+04	5.89E+04	1.12E+05	1.93E+04	1.25E+05	1.53E+06	2.00E+04
FN30	*Std*	3.15E+04	5.40E+04	1.41E+05	4.99E+03	4.23E+04	4.22E+05	6.37E+03
	Rank	3	4	5	1	6	7	2
Average rank		3.30	4.83	3.70	2.10	4.10	6.07	2.47
Overall rank		3	6	4	1	5	7	2

significant differences between the original SCA and the proposed $ImpSCA_4$ for all feature selection datasets.

Fig. 8 shows the comparison results of the SCA and ImpSCAs in terms of classification accuracy, fitness, and selection ratio. From the subfigure (a), we see that the proposed $ImpSCA_4$ version outperforms other versions of ImpSCA and the original SCA according to the accuracy metrics. In terms of the fitness values, it is observed that the results obtained from $ImpSCA_4$ are closer to the center of the polar circle in most feature selection datasets. The figure regarding the feature selection ratio shows that the original SCA has better than the proposed ImpSCAs, but this did not increase the classification accuracy.

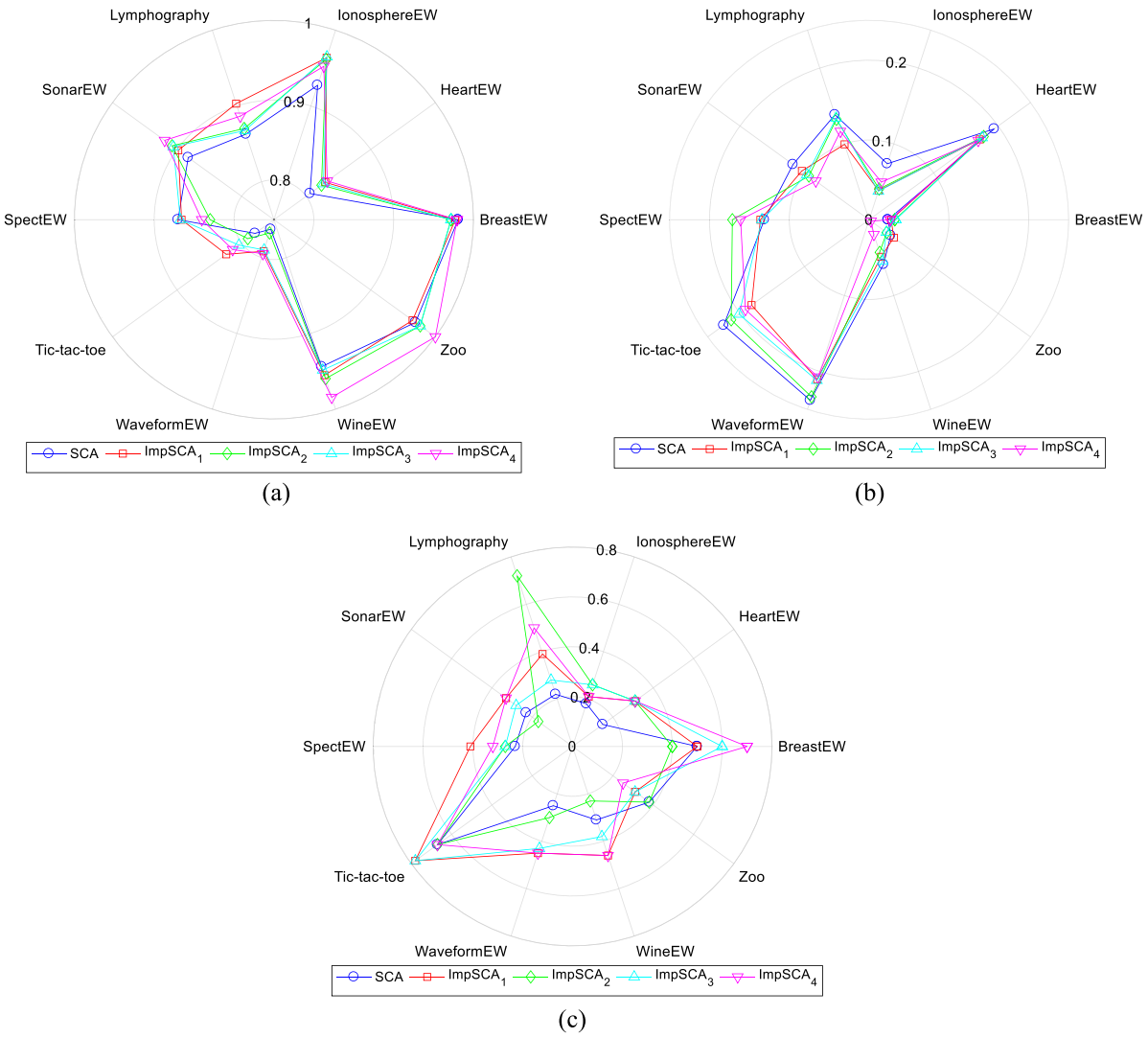


Fig. 8. Comparison results between SCA and ImpSCAs in terms of classification accuracy (a), fitness (b), and selection ratio (c). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5.3.2. Comparison with other metaheuristic algorithms

In this part, the proposed ImpSCA versions are compared with some metaheuristic algorithms for the classification datasets used in this study. Table 17 summarizes the comparison results on the average classification accuracy of the ImpSCA and different methods. In this table, we used the performances of the binary Gray Wolf Optimizer versions (bGWO$_1$-bGWO$_2$) (Emary et al., 2016b), Butterfly Optimization Algorithm (BOA) (Arora and Singh, 2019) and Binary Grasshopper Optimization Algorithm with mutation operator (BGOA-M) (Mafarja et al., 2019) for these classification datasets.

From the comparison results in Table 17, it seems that the ImpSCA$_4$ version has the best performance according to the average accuracy values for all dataset. The ImpSCA$_1$ version is the second with the best performance in terms of the average metric of accuracy values for all dataset. For IonosphereEW and SpectEW dataset, ImpSCA$_3$ version has the best average accuracy value among the other heuristic algorithms and ImpSCA versions. Looking at all the results, it is seen that three versions of the proposed ImpSCAs have more successful classification accuracy values than BOA, BGOA-M and binary GWO versions except of the ImpSCA$_2$.

The comparison and statistical results obtained for the feature selection problem dataset are similar to the results obtained in the CEC2014 benchmark problems presented in the previous subsection in terms of the ranking of the algorithms. Comparing the accuracy values obtained

Table 11
List of datasets used in the experiments.

	Dataset	No. of attributes	No. of objects	No. of classes
1	BreastEW	10	569	2
2	HeartEW	13	270	2
3	IonosphereEW	34	351	2
4	Lymphography	18	148	2
5	SonarEW	60	208	2
6	SpectEW	22	267	2
7	Tic-tac-toe	9	958	2
8	WaveformEW	40	5000	3
9	WineEW	13	178	3
10	Zoo	16	101	6

Table 12
Parameter settings for feature selection experiments.

Parameter	Value
Population size	5
Number of iterations	100
Problem dimension	Number of features
Number of runs	5
α parameter in the fitness function	0.99
Search domain	[0,1]

Table 13
Best classification accuracy and selected attributes for SCA and ImpSCAs.

Dataset	Accuracy					Number of selected features				
	SCA	$ImpSCA_1$	$ImpSCA_2$	$ImpSCA_3$	$ImpSCA_4$	SCA	$ImpSCA_1$	$ImpSCA_2$	$ImpSCA_3$	$ImpSCA_4$
BreastEW	**0.9854**	0.9795	0.9737	0.9737	0.9825	5	5	**4**	6	7
HeartEW	0.8148	0.8370	**0.8593**	0.8296	0.8370	**2**	4	4	4	4
IonosphereEW	0.9318	0.9716	**0.9773**	**0.9773**	0.9545	**6**	7	9	9	7
Lymphography	0.8919	**0.9452**	0.8919	0.8784	0.9189	**4**	7	13	5	9
SonarEW	0.9135	0.9135	0.9231	0.9327	**0.9519**	14	20	**10**	17	20
SpectEW	0.8806	0.8806	0.8433	**0.8881**	0.8507	**5**	9	6	6	7
Tic-tac-toe	0.7891	**0.8246**	0.8038	0.8058	0.8205	**6**	7	**6**	7	**6**
WaveformEW	0.7748	0.7952	0.7892	0.7952	**0.8024**	**10**	18	12	17	18
WineEW	0.9551	0.9663	0.9775	0.9551	**0.9888**	4	6	**3**	5	6
Zoo	0.9787	0.9804	0.9804	0.9804	**1.0000**	6	5	6	5	**4**

Table 14
Mean, best and worst fitness values obtained from the SCA and ImpSCAs.

Dataset		SCA	$ImpSCA_1$	$ImpSCA_2$	$ImpSCA_3$	$ImpSCA_4$
BreastEW	*Best*	**0.020**	0.025	0.030	0.032	0.024
	Mean	**0.024**	0.028	0.032	0.033	0.027
	Worst	**0.030**	0.034	0.034	0.035	0.031
HeartEW	*Best*	0.185	0.164	**0.142**	0.172	0.164
	Mean	0.194	0.171	0.178	0.175	**0.169**
	Worst	0.206	**0.178**	0.222	0.181	0.186
IonosphereEW	*Best*	0.069	0.030	**0.025**	**0.025**	0.047
	Mean	0.074	0.038	0.041	**0.038**	0.050
	Worst	0.081	**0.047**	0.052	0.049	0.053
Lymphography	*Best*	0.109	**0.058**	0.114	0.123	0.085
	Mean	0.139	**0.099**	0.132	0.134	0.117
	Worst	0.167	0.154	0.150	0.163	**0.137**
SonarEW	*Best*	0.088	0.089	0.078	0.070	**0.051**
	Mean	0.118	0.104	0.093	0.096	**0.083**
	Worst	0.154	0.126	**0.106**	0.108	0.107
SpectEW	*Best*	0.121	0.122	0.158	**0.113**	0.151
	Mean	**0.131**	0.136	0.171	0.134	0.161
	Worst	0.158	**0.150**	0.188	0.151	0.180
Tic-tac-toe	*Best*	0.215	**0.181**	0.201	0.200	0.184
	Mean	0.225	**0.182**	0.214	0.201	0.192
	Worst	0.256	**0.184**	0.232	0.207	0.202
WaveformEW	*Best*	0.225	0.207	0.212	0.207	**0.200**
	Mean	0.238	0.210	0.234	0.212	**0.207**
	Worst	0.255	0.213	0.255	0.216	**0.212**
WineEW	*Best*	0.048	0.038	0.025	0.048	**0.014**
	Mean	0.059	0.049	0.043	0.055	**0.019**
	Worst	0.080	0.060	0.058	0.059	**0.026**
Zoo	*Best*	0.025	0.023	0.023	0.023	**0.003**
	Mean	0.033	0.038	0.027	0.027	**0.003**
	Worst	0.061	0.061	0.041	0.042	**0.004**

Table 15
Mean, best and worst accuracy values obtained from the SCA and ImpSCAs.

Dataset		SCA	$ImpSCA_1$	$ImpSCA_2$	$ImpSCA_3$	$ImpSCA_4$
BreastEW	*Best*	**0.985**	0.980	0.974	0.974	0.983
	Mean	**0.981**	0.977	0.972	0.972	0.980
	Worst	**0.974**	0.971	0.971	0.971	**0.974**
HeartEW	*Best*	0.815	0.837	**0.859**	0.830	0.837
	Mean	0.806	0.830	0.824	0.827	**0.833**
	Worst	0.793	**0.822**	0.778	**0.822**	0.815
IonosphereEW	*Best*	0.932	0.972	**0.977**	**0.977**	0.955
	Mean	0.927	**0.964**	0.961	0.965	0.952
	Worst	0.921	**0.955**	0.949	**0.955**	0.949
Lymphography	*Best*	0.892	**0.945**	0.892	0.878	0.919
	Mean	0.863	**0.903**	0.870	0.868	0.887
	Worst	0.833	0.847	0.851	0.838	**0.865**
SonarEW	*Best*	0.914	0.914	0.923	0.933	**0.952**
	Mean	0.883	0.898	0.908	0.906	**0.920**
	Worst	0.846	0.875	**0.894**	**0.894**	**0.894**
SpectEW	*Best*	0.881	0.881	0.843	**0.888**	0.851
	Mean	**0.870**	0.866	0.830	0.867	0.840
	Worst	0.843	**0.851**	0.813	**0.851**	0.821
Tic-tac-toe	*Best*	0.789	**0.825**	0.804	0.806	0.821
	Mean	0.780	**0.824**	0.790	0.804	0.814
	Worst	0.747	**0.821**	0.772	0.798	0.804
WaveformEW	*Best*	0.775	0.795	0.789	0.795	**0.802**
	Mean	0.763	0.792	0.768	0.790	**0.795**
	Worst	0.746	0.789	0.747	0.784	**0.791**
WineEW	*Best*	0.955	0.966	0.978	0.955	**0.989**
	Mean	0.944	0.955	0.960	0.948	**0.984**
	Worst	0.921	0.944	0.944	0.944	**0.978**
Zoo	*Best*	0.979	0.980	0.980	0.980	**1.000**
	Mean	0.969	0.965	0.977	0.977	**1.000**
	Worst	0.940	0.941	0.961	0.961	**1.000**

for the same dataset of different metaheuristic algorithms with the proposed ImpSCAs, it is understood that $ImpSCA_1$ and $ImpSCA_4$ versions are in the first two ranks. The reason for this is due to the search ability of the proposed update mechanisms in the exploration and exploitation phases of both algorithms, as we mentioned before. At the same time, better results were obtained in feature selection problems thanks to mutation, greedy selection and boundary checking mechanisms added to the original SCA structure. Although the selection mechanism added to the algorithm seems to reduce the diversity in the population, the results show that the search agents with close to each other avoid the local optimal solutions and approach the global solution point thanks to the update mechanisms developed.

Table 16
P-values of the Wilcoxon rank-sum test with 5% significance for fitness values obtained by SCA and ImpSCAs in feature selection problem dataset.

Dataset	$ImpSCA_1$ versus SCA	$ImpSCA_2$ versus SCA	$ImpSCA_3$ versus SCA	$ImpSCA_4$ versus SCA
BreastEW	**3.89E−01**	2.38E−02	7.94E−03	**3.89E−01**
HeartEW	7.94E−03	**5.24E−01**	7.94E−03	2.38E−02
IonosphereEW	7.94E−03	7.94E−03	7.94E−03	7.94E−03
Lymphography	**2.06E−01**	**6.51E−01**	**5.00E−01**	**1.51E−01**
SonarEW	**4.21E−01**	4.76E−02	**9.52E−02**	3.17E−02
SpectEW	**2.78E−01**	1.59E−02	**7.14E−01**	4.76E−02
Tic-tac-toe	7.94E−03	**3.81E−01**	7.94E−03	7.94E−03
WaveformEW	7.94E−03	**1.00E+00**	7.94E−03	7.94E−03
WineEW	**4.44E−01**	**1.98E−01**	**1.00E+00**	7.94E−03
Zoo	**6.59E−01**	**8.73E−02**	**8.73E−02**	7.94E−03

6. Conclusion and future works

In this paper, the four novel versions of the original SCA were proposed to improve the exploration and exploitation ability of the algorithm. The basic idea behind these proposed ImpSCA versions is based on the innovations on the updating mechanism of the original SCA. The first version of ImpSCAs uses a combination of the updating equations with sine and cosine functions. In the second approach, the position of a search agent selected randomly from the search space is utilized instead of the destination position in the updating mechanism.

Table 17
Comparison results between the ImpSCA versions and other metaheuristics.

Dataset	BOA	BGOA-M	$bGWO_1$	$bGWO_2$	$ImpSCA_1$	$ImpSCA_2$	$ImpSCA_3$	$ImpSCA_4$
BreastEW	0.951	0.970	0.924	0.935	0.977	0.972	0.972	0.980
HeartEW	0.801	0.836	0.776	0.776	0.830	0.824	0.827	0.833
IonosphereEW	0.902	0.946	0.807	0.834	0.964	0.961	0.965	0.952
Lymphography	0.822	0.912	0.744	0.700	0.903	0.870	0.868	0.887
SonarEW	0.885	0.915	0.731	0.729	0.898	0.908	0.906	0.920
SpectEW	0.815	0.826	0.820	0.822	0.866	0.830	0.867	0.840
Tic-tac-toe	0.774	0.791	0.728	0.727	0.824	0.790	0.804	0.814
WaveformEW	0.729	0.751	0.786	0.789	0.792	0.768	0.790	0.795
WineEW	0.972	0.989	0.930	0.920	0.955	0.960	0.948	0.984
Zoo	0.961	0.958	0.879	0.879	0.965	0.977	0.977	1.000
Average	0.861	0.889	0.813	0.811	0.897	0.886	0.892	0.901
Rank	6	4	7	8	2	5	3	1

BOA: Butterfly Optimization Algorithm, BGOA-M: Binary Grasshopper Optimization Algorithm with Mutation, $bGWO_1$: binary Grey Wolf Optimizer version 1, $bGWO_2$: binary Grey Wolf Optimizer version 2.

The third version of ImpSCA is based on the weighted average of the positions obtained by the updating mechanism of the original SCA. In the last approach, a novel mechanism is presented, similar to the crossover operator in the genetic algorithm. In addition to these innovations in update mechanisms, mutation operator, boundary checking mechanism, and simple selection mechanism have been added to all of these proposed versions.

First, the analyses of the ImpSCAs were realized, such as convergence, search history, trajectory, and average distance. These analyzes revealed the advantages and disadvantages of the proposed structures for ImpSCA versions compared to the original SCA structure. The ImpSCA versions were evaluated for 30 benchmark functions from CEC 2014 and adapted to the feature selection problems from UCI data repository. The benchmark results show that $ImpSCA_1$ is the first according to the statistical metrics, and the second algorithm is $ImpSCA_4$ version. The results obtained for feature selection datasets demonstrated the superiority of the $ImpSCA_4$ over the other ImpSCA versions and the original SCA. Wilcoxon rank-sum statistical test results show that the ImpSCA versions (except for the 3rd version of ImpSCA) have significant differences with better convergence than the original SCA.

The results obtained with the proposed ImpSCA versions show us that these structures can be good alternatives for solving different real-world optimization problems. ImpSCAs can be easily adapted and utilized in optimizing the model parameters of models such as artificial neural networks and fuzzy logic used in the field of artificial intelligence. Also, combinatorial optimization problems can be solved by changing the search agent encoding structure used in the proposed ImpSCA versions. Pareto optimal based ImpSCA versions can be developed for multi-objective optimization problems. As a result, the proposed ImpSCAs will provide opportunities for future studies to researchers working in different fields.

In these future studies, $ImpSCA_4$ and $ImpSCA_1$ versions, which have better performance among the proposed versions, can be applied to real-world problems, such as robot path planning, parallel machine scheduling, economic dispatching, facility layout design, vehicle routing, and bin packing problems. As the next step, the versions of ImpSCA can be hybridized with other algorithms. Finally, multi-objective versions of ImpSCAs can be proposed to solve multi-objective optimization problems.

CRediT authorship contribution statement

Gizem Ataç Kale: Methodology, Software, Writing – original draft. **Uğur Yüzgeç:** Conceptualization, Software, Validation, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Abd Elaziz, M., Oliva, D., Xiong, S., 2017. An improved Opposition-Based Sine Cosine Algorithm for global optimization. Expert Syst. Appl. 90, 484–500. http://dx.doi.org/10.1016/j.eswa.2017.07.043.

Abdollahzadeh, B., Gharehchopogh, F.S., Mirjalili, S., 2021a. African vultures optimization algorithm: A new nature-inspired metaheuristic algorithm for global optimization problems. Comput. Ind. Eng. 158, 107408. http://dx.doi.org/10.1016/j.cie.2021.107408.

Abdollahzadeh, B., Gharehchopogh, F.S., Mirjalili, S., 2021b. Artificial gorilla troops optimizer: A new nature-inspired metaheuristic algorithm for global optimization problems. Int. J. Intell. Syst. 1–72. http://dx.doi.org/10.1002/int.22535.

Acuña, E., Rodriguez, C., 2004. The treatment of missing values and its effect on classifier accuracy. In: Classification, Clustering, and Data Mining Applications. pp. 639–647. http://dx.doi.org/10.1007/978-3-642-17103-1_60.

Aggarwal, C.C., 2014. Educational and software resources for data classification. In: Data Classification: Algorithms and Applications. pp. 657–665. http://dx.doi.org/10.1201/b17320.

Alelyani, Salem, Tang, Jiliang, Liu, Huan, 2013. Feature selection for clustering: A review. Data Clust. Algorithms Appl. 29, 110–121. http://dx.doi.org/10.1201/9781315373515-2.

Arora, S., Singh, S., 2019. Butterfly optimization algorithm: a novel approach for global optimization. Soft Comput. 23, 715–734. http://dx.doi.org/10.1007/s00500-018-3102-4.

Atashpaz-Gargari, E., Lucas, C., 2007. Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition. In: 2007 IEEE Congress on Evolutionary Computation, CEC 2007. pp. 4661–4667. http://dx.doi.org/10.1109/CEC.2007.4425083.

Bairathi, D., Gopalani, D., 2020. A novel swarm intelligence based optimization method: Harris Hawk optimization. In: Advances in Intelligent Systems and Computing. pp. 832–842. http://dx.doi.org/10.1007/978-3-030-16660-1_81.

Beyer, H., Sendhoff, B., 2008. Covariance matrix adaptation revisited–the CMSA evolution strategy–. In: Parallel Probl. Solving from Nature–PPSN X. pp. 123–132. http://dx.doi.org/10.1007/978-3-540-87700-4_13.

Bhattacharjee, K.K., Sarmah, S.P., 2014. Shuffled frog leaping algorithm and its application to 0/1 knapsack problem. Appl. Soft Comput. J. 19, 252–263. http://dx.doi.org/10.1016/j.asoc.2014.02.010.

Bianchi, L., Dorigo, M., Gambardella, L.M., Gutjahr, W.J., 2009. A survey on metaheuristics for stochastic combinatorial optimization. Nat. Comput. 8, 239–287. http://dx.doi.org/10.1007/s11047-008-9098-4.

Boussaïd, I., Lepagnot, J., Siarry, P., 2013. A survey on optimization metaheuristics. In: Information Sciences. pp. 82–117. http://dx.doi.org/10.1016/j.ins.2013.02.041.

Chandrashekar, G., Sahin, F., 2014. A survey on feature selection methods. Comput. Electr. Eng. 40, 16–28. http://dx.doi.org/10.1016/j.compeleceng.2013.11.024.

Coello Coello, C.A., Becerra, R.L., 2004. Efficient evolutionary optimization through the use of a cultural algorithm. Eng. Optim. 36, 219–236. http://dx.doi.org/10.1080/03052150410001647966.

Dorigo, M., Birattari, M., Stutzle, T., 2006. Ant colony optimization. IEEE Comput. Intell. Mag. 1, 28–39. http://dx.doi.org/10.1109/MCI.2006.329691.

Dorigo, M., Maniezzo, V., Colorni, A., 1996. Ant system: Optimization by a colony of cooperating agents. IEEE Trans. Syst. Man Cybern. B 26, 29–41. http://dx.doi.org/10.1109/3477.484436.

Emary, E., Zawbaa, H.M., Hassanien, A.E., 2016a. Binary ant lion approaches for feature selection. Neurocomputing 213, 54–65. http://dx.doi.org/10.1016/j.neucom.2016.03.101.

Emary, E., Zawbaa, H.M., Hassanien, A.E., 2016b. Binary grey wolf optimization approaches for feature selection. Neurocomputing 172, 371–381. http://dx.doi.org/10.1016/j.neucom.2015.06.083.

Eusuff, M., Lansey, K., Pasha, F., 2006. Shuffled frog-leaping algorithm: A memetic meta-heuristic for discrete optimization. Eng. Optim. 38, 129–154. http://dx.doi.org/10.1080/03052150500384759.

Geem, Z.W., Kim, J.H., Loganathan, G.V., 2001. A new heuristic optimization algorithm: Harmony search. Simulation 76, 60–68. http://dx.doi.org/10.1177/003754970107600201.

Gendreau, M., Iori, M., Laporte, G., Martello, S., 2006. A tabu search algorithm for a routing and container loading problem. Transp. Sci. 40, 342–350. http://dx.doi.org/10.1287/trsc.1050.0145.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison Wesley, http://dx.doi.org/10.1007/s10589-009-9261-6.

Gupta, S., Deep, K., Engelbrecht, A.P., 2020. A memory guided sine cosine algorithm for global optimization. Eng. Appl. Artif. Intell. 93, 1–19. http://dx.doi.org/10.1016/j.engappai.2020.103718.

Heidari, A.A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., Chen, H., 2019. Harris hawks optimization: Algorithm and applications. Future Gener. Comput. Syst. 97, 849–872. http://dx.doi.org/10.1016/j.future.2019.02.028.

Ho, Y.C., Pepyne, D.L., 2002. Simple explanation of the no-free-lunch theorem and its implications. J. Optim. Theory Appl. 115, 549–570. http://dx.doi.org/10.1023/A:1021251113462.

Holland, J.H., 1975. Adaptation in Natural and Artificial Systems. Ann Arbor MI Univ. Michigan Press, Ann Arbor, p. 183. http://dx.doi.org/10.1137/1018105.

Hosseini, S., Al Khaled, A., 2014. A survey on the imperialist competitive algorithm metaheuristic: Implementation in engineering domain and directions for future research. Appl. Soft Comput. J. http://dx.doi.org/10.1016/j.asoc.2014.08.024.

Iguyon, I., Elisseeff, A., 2003. An introduction to variable and feature selection. J. Mach. Learn. Res. http://dx.doi.org/10.1162/153244303322753616.

Iscan, H., Gunduz, M., 2014. Parameter analysis on fruit fly optimization algorithm. J. Comput. Commun. 2, 137–141. http://dx.doi.org/10.1109/SITIS.2015.55.

Issa, M., Hassanien, A.E., Oliva, D., Helmi, A., Ziedan, I., Alzohairy, A., 2018. ASCA-PSO: Adaptive sine cosine optimization algorithm integrated with particle swarm for pairwise local sequence alignment. Expert Syst. Appl. 99, 56–70. http://dx.doi.org/10.1016/j.eswa.2018.01.019.

Jensen, R., Shen, Q., 2008. Computational intelligence and feature selection: Rough and fuzzy approaches. In: Computational Intelligence and Feature Selection: Rough and Fuzzy Approaches. http://dx.doi.org/10.1002/9780470377888.

Jović, A., Brkić, K., Bogunović, N., 2015. A review of feature selection methods with applications. In: 2015 38th Int. Conv. Inf. Commun. Technol. Electron. Microelectron. MIPRO 2015 - Proc.. pp. 1200–1205. http://dx.doi.org/10.1109/MIPRO.2015.7160458.

Karaboga, D., 2005. An Idea Based on Honey Bee Swarm for Numerical Optimization. Tech. Rep. TR06, Erciyes Univ., p. 10.

Karaboga, D., Akay, B., 2009. A comparative study of artificial bee colony algorithm. Appl. Math. Comput. 214, 108–132. http://dx.doi.org/10.1016/j.amc.2009.03.090.

Karaboga, D., Basturk, B., 2008. On the performance of artificial bee colony (ABC) algorithm. Appl. Soft Comput. J. 8, 687–697. http://dx.doi.org/10.1016/j.asoc.2007.05.007.

Kaveh, A., Ghazaan, M.I., 2017. Enhanced whale optimization algorithm for sizing optimization of skeletal structures. Mech. Based Des. Struct. Mach. 45, 345–362. http://dx.doi.org/10.1080/15397734.2016.1213639.

Kennedy, J., Eberhart, R., 1995. Particle swarm optimization. In: Neural Networks, 1995. Proceedings. IEEE International Conference on. pp. 1942–1948. http://dx.doi.org/10.1109/ICNN.1995.488968.

Khushaba, R.N., Al-Ani, A., Al-Jumaily, A., 2008. Differential evolution based feature subset selection. In: Proceedings - International Conference on Pattern Recognition. pp. 1–4. http://dx.doi.org/10.1109/icpr.2008.4761255.

Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., 1983. Optimization by simulated annealing. Science (80-.) 220, 671–680. http://dx.doi.org/10.1126/science.220.4598.671.

Kothari, V., Anuradha, J., Shah, S., Mittal, P., 2012. A survey on particle swarm optimization in feature selection. In: Communications in Computer and Information Science. pp. 192–201. http://dx.doi.org/10.1007/978-3-642-29216-3_22.

Liang, J.J., Qu, B.Y., Suganthan, P.N., 2014. Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization, 2014 IEEE Congress on Evolutionary Computation. http://dx.doi.org/10.11884/HPLPB201729.170238.

Lichman, M., 2013. UCI machine learning repository. [http://archive.ics.uci.edu/ml]. UCI Machine Learning Repository.

Ma, H., Simon, D., 2011. Blended biogeography-based optimization for constrained optimization. Eng. Appl. Artif. Intell. 24, 517–525. http://dx.doi.org/10.1016/j.engappai.2010.08.005.

Mafarja, M., Aljarah, I., Faris, H., Hammouri, A.I., Al-Zoubi, A.M., Mirjalili, S., 2019. Binary grasshopper optimisation algorithm approaches for feature selection problems. Expert Syst. Appl. 117, 267–286. http://dx.doi.org/10.1016/j.eswa.2018.09.015.

Mafarja, M., Aljarah, I., Heidari, A.A., Faris, H., Fournier-Viger, P., Li, X., Mirjalili, S., 2018. Binary dragonfly optimization for feature selection using time-varying transfer functions. Knowl.-Based Syst. 161, 185–204. http://dx.doi.org/10.1016/j.knosys.2018.08.003.

Mafarja, M.M., Mirjalili, S., 2018a. Hybrid binary ant lion optimizer with rough set and approximate entropy reducts for feature selection. Soft Comput. 23, 1–17. http://dx.doi.org/10.1007/s00500-018-3282-y.

Mafarja, M., Mirjalili, S., 2018b. Whale optimization approaches for wrapper feature selection. Appl. Soft Comput. J. 62, 441–453. http://dx.doi.org/10.1016/j.asoc.2017.11.006.

Mavridou, T.D., Pardalos, P.M., 1997. Simulated annealing and genetic algorithms for the facility layout problem: A survey. Comput. Optim. Appl. 7, 111–126. http://dx.doi.org/10.1023/A:1008623913524.

Mehrabian, A.R., Lucas, C., 2006. A novel numerical optimization algorithm inspired from weed colonization. Ecol. Inform. 1, 355–366. http://dx.doi.org/10.1016/j.ecoinf.2006.07.003.

Mirjalili, S., 2015a. How effective is the Grey Wolf optimizer in training multi-layer perceptrons. Appl. Intell. 43, 150–161. http://dx.doi.org/10.1007/s10489-014-0645-7.

Mirjalili, S., 2015b. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. Knowl.-Based Syst. 89, 228–249. http://dx.doi.org/10.1016/j.knosys.2015.07.006.

Mirjalili, S., 2016a. Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Comput. Appl. 27, 1053–1073. http://dx.doi.org/10.1007/s00521-015-1920-1.

Mirjalili, S., 2016b. SCA: A Sine Cosine Algorithm for solving optimization problems. Knowl.-Based Syst. 96, 120–133. http://dx.doi.org/10.1016/j.knosys.2015.12.022.

Mirjalili, S., Lewis, A., 2016. The whale optimization algorithm. Adv. Eng. Softw. 95, 51–67. http://dx.doi.org/10.1016/j.advengsoft.2016.01.008.

Mirjalili, S., Mirjalili, S.M., Lewis, A., 2014. Grey wolf optimizer. Adv. Eng. Softw. 69, 46–61. http://dx.doi.org/10.1016/j.advengsoft.2013.12.007.

Nayak, D.R., Dash, R., Majhi, B., Wang, S., 2018. Combining extreme learning machine with modified sine cosine algorithm for detection of pathological brain. Comput. Electr. Eng. 68, 366–380. http://dx.doi.org/10.1016/j.compeleceng.2018.04.009.

Nenavath, H., Jatoth, R.K., 2018. Hybridizing sine cosine algorithm with differential evolution for global optimization and object tracking. Appl. Soft Comput. J. 62, 1019–1043. http://dx.doi.org/10.1016/j.asoc.2017.09.039.

Nowicki, E., Smutnicki, C., 1996. A fast tabu search algorithm for the permutation flow-shop problem. European J. Oper. Res. 91, 160–175. http://dx.doi.org/10.1016/0377-2217(95)00037-2.

Pan, W.T., 2012. A new fruit fly optimization algorithm: Taking the financial distress model as an example. Knowl.-Based Syst. 26, 69–74. http://dx.doi.org/10.1016/j.knosys.2011.07.001.

Poli, R., Kennedy, J., Blackwell, T., 2007. Particle swarm optimization. Swarm Intell. 1, 33–57. http://dx.doi.org/10.1007/s11721-007-0002-0.

Qu, C., Zeng, Z., Dai, J., Yi, Z., He, W., 2018. A modified Sine-Cosine Algorithm based on neighborhood search and greedy Levy mutation. Comput. Intell. Neurosci. 2018, 1–19. http://dx.doi.org/10.1155/2018/4231647.

Rao, H., Shi, X., Rodrigue, A.K., Feng, J., Xia, Y., Elhoseny, M., Yuan, X., Gu, L., 2019. Feature selection based on artificial bee colony and gradient boosting decision tree. Appl. Soft Comput. J. 74, 634–642. http://dx.doi.org/10.1016/j.asoc.2018.10.036.

Rashedi, E., Nezamabadi-pour, H., Saryazdi, S., 2009. GSA: A gravitational search algorithm. Inf. Sci. (Ny) 179, 2232–2248. http://dx.doi.org/10.1016/j.ins.2009.03.004.

Sabri, N.M., Puteh, M., Mahmood, M.R., 2013. A review of gravitational search algorithm. Int. J. Adv. Soft Comput. Appl. 5.

Saremi, S., Mirjalili, S., Lewis, A., 2017. Grasshopper optimisation algorithm: Theory and application. Adv. Eng. Softw. 105, 30–47. http://dx.doi.org/10.1016/j.advengsoft.2017.01.004.

Sayed, G.I., Hassanien, A.E., Azar, A.T., 2019. Feature selection via a novel chaotic crow search algorithm. Neural Comput. Appl. 31, 171–188. http://dx.doi.org/10.1007/s00521-017-2988-6.

Sayed, G.I., Khoriba, G., Haggag, M.H., 2018. A novel chaotic salp swarm algorithm for global optimization and feature selection. Appl. Intell. 48, 3462–3481. http://dx.doi.org/10.1007/s10489-018-1158-6.

Shunmugapriya, P., Kanmani, S., 2012. Artificial bee colony approach for optimizing feature selection. Int. J. Comput. Sci. Issues 9, 432–438.

Sikdar, U.K., Ekbal, A., Saha, S., 2012. Differential evolution based feature selection and classifier ensemble for named entity recognition. In: 24th International Conference on Computational Linguistics - Proceedings of COLING 2012: Technical Papers. pp. 2475–2490.

Simon, D., 2008. Biogeography-based optimization. IEEE Trans. Evol. Comput. 12, 702–713. http://dx.doi.org/10.1109/TEVC.2008.919004.

Sree Ranjini, S.R., Murugan, S., 2017. Memory based Hybrid Dragonfly Algorithm for numerical optimization problems. Expert Syst. Appl. 83, 63–78. http://dx.doi.org/10.1016/j.eswa.2017.04.033.

Sreejith, S., Chandrasekaran, K., Simon, S.P., 2009. Touring ant colony optimization technique for optimal power flow incorporating thyristor controlled series compensator. In: 2009 World Congress on Nature and Biologically Inspired Computing, NABIC 2009 - Proceedings. pp. 1127–1132. http://dx.doi.org/10.1109/NABIC.2009.5393815.

Storn, R., Price, K., 1997. Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. 11, 341–359. http://dx.doi.org/10.1023/A:1008202821328.

Suid, M.H., Ahmad, M.A., Ismail, M.R.T.R., Ghazali, M.R., Irawan, A., Tumari, M.Z., 2018. An improved Sine Cosine Algorithm for solving optimization problems. In: Proc. - 2018 IEEE Conf. Syst. Process Control. ICSPC 2018. pp. 209–213. http://dx.doi.org/10.1109/SPC.2018.8703982.

Tiwari, A.K., 2017. Introduction to machine learning. In: Ubiquitous Mach. Learn. its Appl.. http://dx.doi.org/10.4018/978-1-5225-2545-5.ch001.

Willjuice Iruthayarajan, M., Baskar, S., 2010. Covariance matrix adaptation evolution strategy based design of centralized PID controller. Expert Syst. Appl. 37, 5775–5781. http://dx.doi.org/10.1016/j.eswa.2010.02.031.

Wolpert, D.H., Macready, W.G., 1997. No free lunch theorems for optimization. IEEE Trans. Evol. Comput. 1, 67–82. http://dx.doi.org/10.1109/4235.585893.

Xu, Y., Chen, H., Heidari, A.A., Luo, J., Zhang, Q., Zhao, X., Li, C., 2019. An efficient chaotic mutative moth-flame-inspired optimizer for global optimization tasks. Expert Syst. Appl. 129, 135–155. http://dx.doi.org/10.1016/j.eswa.2019.03.043.

Xue, B., Zhang, M., Browne, W.N., 2014. Particle swarm optimisation for feature selection in classification: Novel initialisation and updating mechanisms. Appl. Soft Comput. J. 18, 261–276. http://dx.doi.org/10.1016/j.asoc.2013.09.018.

Yang, X.-S., 2009. Cuckoo search via Lévy flights. In: 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC). pp. 210–214. http://dx.doi.org/10.1109/NABIC.2009.5393690.

Yang, X.S., 2009a. Firefly algorithms for multimodal optimization. In: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp. 169–178. http://dx.doi.org/10.1007/978-3-642-04944-6_14.

Yang, X.S., 2009b. Harmony search as a metaheuristic algorithm. Stud. Comput. Intell. http://dx.doi.org/10.1007/978-3-642-00185-7_1.

Yang, X.S., 2010. Firefly algorithm, Lévy flights and global optimization. In: Research and Development in Intelligent Systems XXVI: Incorporating Applications and Innovations in Intelligent Systems XVII. pp. 1–10. http://dx.doi.org/10.1007/978-1-84882-983-1-15.

Yang, X.S., Deb, S., 2014. Cuckoo search: Recent advances and applications. Neural Comput. Appl. http://dx.doi.org/10.1007/s00521-013-1367-1.

Yüzgeç, U., 2010. Performance comparison of differential evolution techniques on optimization of feeding profile for an industrial scale baker's yeast fermentation process. ISA Trans. 49, 167–176. http://dx.doi.org/10.1016/j.isatra.2009.10.006.

Zhang, X., Niu, Y., Cui, G., Wang, Y., 2010. A modified invasive weed optimization with crossover operation. In: Proceedings of the 8th World Congress on Intelligent Control and Automation. pp. 11–14. http://dx.doi.org/10.1109/WCICA.2010.5553805.