Applied Soft Computing Journal 85 (2019) 105734

Contents lists available at ScienceDirect

Applied Soft Computing Journal

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

The spherical search algorithm for bound-constrained global optimization problems

Abhishek Kumar [a], Rakesh Kumar Misra [a], Devender Singh [a], Sujeet Mishra [b], Swagatam Das [c,*]

[a] Department of Electrical Engineering, Indian Institute of Technology (BHU), Varanasi, Varanasi, 221005, India
[b] Chief Design Engineer (Electrical), Diesel Locomotive Works, Varanasi, India
[c] Electronics and Communication Sciences Unit, Indian Statistical Institute, Kolkata, India

ARTICLE INFO

Article history:
Received 17 February 2019
Received in revised form 22 July 2019
Accepted 25 August 2019
Available online 14 September 2019

Keywords:
Spherical search algorithm
Real-life optimization problems
Bound constrained optimization problem
Optimization algorithm
Global optimization

ABSTRACT

In this paper, a new optimization algorithm called Spherical Search (SS) is proposed to solve the bound-constrained non-linear global optimization problems. The main operations of SS are the calculation of spherical boundary and generation of new trial solution on the surface of the spherical boundary. These operations are mathematically modeled with some more basic level operators: Initialization of solution, greedy selection and parameter adaptation, and are employed on the 30 black-box bound constrained global optimization problems. This study also analyzes the applicability of the proposed algorithm on a set of real-life optimization problems. Meanwhile, to show the robustness and proficiency of SS, the obtained results of the proposed algorithm are compared with the results of other well-known optimization algorithms and their advanced variants: Particle Swarm Optimization (PSO), Differential Evolution (DE), and Covariance Matrix Adapted Evolution Strategy (CMA-ES). The comparative analysis reveals that the performance of SS is quite competitive with respect to the other peer algorithms.

1. Introduction

For over the last few decades, complexity of real-life optimization problems has been rapidly increasing with the advent of latest technologies. Solving these optimization problems is an essential component of any engineering design problem. So far numerous optimization techniques have been proposed and adapted to provide the optimal solutions for different optimization problems. According to the nature of operators, these algorithms can be classified into two classes: Deterministic techniques and Meta-heuristics. In deterministic techniques, the solution of the previous iteration is used to determine the updated solution for the current iteration. Therefore, in the case of deterministic techniques, the choice of the initial solution influences the final solution. Furthermore, the solutions can be a victim of easily getting trapped into the local optima. Consequently, deterministic techniques are less efficient and less effective tools for solving multi-modal, highly complex, and high-dimensional optimization problems. As an alternate technique, meta-heuristics have been preferred for solving global optimization problems. A lot of theoretical work on these algorithms have been published in various popular journals thereby mainstreaming meta-heuristics. Principal reasons for the popularity of the said algorithms over deterministic techniques are as follows:

Simplicity- Foremost characteristic of meta-heuristics is the simplicity of theories and techniques. Meta-heuristics are basically inspired by the simple concepts of some biological or physical phenomena.

Flexibility- Meta-heuristics can easily be applied to the different optimization problems with no change or minor changes in the basic structure of the technique. Most of the techniques of meta-heuristics assume the problem as a black box requiring only input and output.

Derivative free- The most important characteristic of these algorithms is a derivative-free mechanism. This means that there is no need for derivatives to solve the real-life optimization problems having complex search space with multiple local minima.

Local optima avoidance- Meta-heuristics have an in-built capability for local optima avoidance. Local optima avoidance is required in the optimization of multi-modal problems. Meta-heuristics hence are preferred over conventional techniques for finding global optima of multi-modal problems.

Researchers have introduced many meta-heuristics. Some of them are popular because of showing good efficiency for most of the real-life optimization problems. The No-Free-Lunch theorem [1] logically proved that there is no universal method, which solves all type of problems efficiently. A particular method may

* Corresponding author.
E-mail address: swagatam.das@isical.ac.in (S. Das).

https://doi.org/10.1016/j.asoc.2019.105734
1568-4946/

Nomenclature

$N \in \mathbb{N}$	number of solutions in population
$k \in \mathbb{N}$	iteration numbers
$D \in \mathbb{N}$	search-space dimension
$\bar{x}_i^{(k)} \in \mathbb{R}^{(D\times 1)}$	i-th solution of population from iteration k
$\bar{y}_i^{(k)} \in \mathbb{R}^{(D\times 1)}$	trial-solution for i-th solution of population from iteration k
$\bar{z}_i^{(k)} \in \mathbb{R}^{(D\times 1)}$	search direction for i-th solution of population at iteration k
$A^{(k)} \in \mathbb{R}^{(D\times D)}$	an orthogonal matrix at iteration k
$B_i^{(k)} = diag(\bar{b}_i^{(k)})$	a binary diagonal matrix at iteration k
$\bar{b}_i^{(k)}$	consist of diagonal elements of matrix $B_i^{(k)}$ at iteration k for i-th solution of population
$\bar{c}^{(k)} \in \mathbb{R}_{>0}^{(N\times 1)}$	a step-size control vector for iteration k
$c_i^{(k)} \in \mathbb{R}_{>0}$	i-th element of step-size control vector $\bar{c}^{(k)}$, a step-size control parameter for i-th solution of population
$diag(\bar{b}_i)$	a square diagonal matrix with the elements of vector $\bar{b}_i$ on the diagonal

show very efficient performance on a particular type of problem, but, the same method may show very poor performance on a different problem.

Meta-heuristics are classified into two classes on the basis of population: one is single-agent based and other is population-based. In single-agent based algorithms, the process starts with a single initial solution. The solution is improved over the course of iterations. In case of population-based algorithms, multiple initial solutions are generated. They are improved over the course of iterations by sharing the group information of search space. The population-based algorithm has the following inherent advantages over single-agent based algorithm:-

1. Multiple solutions share their search information, by which they explore the promising areas of the search space efficiently.
2. Multiple solutions may avoid the local optima of the search space by comparing other solutions.
3. Multiple solutions explore larger search space in one iteration than the single-agent algorithm but they require a greater number of function evaluations.

Some algorithms of meta-heuristics are inspired by natural phenomenons. Based on the source of inspiration, these algorithms can be divided into two classes.

1. Evolutionary algorithms: The evolutionary algorithms (EAs) are motivated by process of natural evolution, such as mutation, crossover, reproduction, and selection. Some popular EAs are Genetic Algorithm (GA) [2], Evolution Strategies (ES) [3], Evolutionary Programming (EP) [4,5], Genetic Programming (GP) [6], and Differential Evolution (DE) [7].
2. Swarm-based algorithms: The swarm-based algorithms (SAs) are usually induced by the mimicking of food-foraging, location-finding, and other intelligent behaviors of social animals or living organism. Particle Swarm optimization (PSO) [8], Ant Colony Optimization (ACO) [9], Artificial Bee Colony (ABC) [10], Grey Wolf Optimizer (GWO) [11], and Bat Algorithm (BA) [12] are some popular example of SAs. Some other recently proposed meta-heuristic optimization algorithms are tabulated in Table 1.

Regardless of the differences among all the classes of meta-heuristics, two major characteristics are common for all: exploration and exploitation. By using exploration characteristic of an algorithm, the individuals try to investigate the promising areas of the search space. However, by using the exploitation characteristic of an algorithm, the individuals try to locally investigate the promising areas. To converge on a global minimum of the problem, a proper balance between exploration and exploitation operators of an algorithm is necessary. For any new meta-heuristic method, proper balancing between the exploration and exploitation operators is a challenging task. A wide spectrum of research on and with such bio-inspired optimization algorithms has been extensively reviewed in [28].

Various parameters are introduced to effectively employ the exploration and exploitation characteristics in an algorithm. These parameters are specifically tuned for a problem to effectively balance the exploration and exploitation characteristics of an algorithm. Thus, the performance of meta-heuristics on a particular type of problem also depends on the parameters they employ. The values of parameters for a particular problem are not known beforehand and one has to employ some rule of thumb or heuristics or trial and error. One of the major contributions of the proposed method is that it is a parameter-free search algorithm.

This work proposes a new swarm-based metaheuristic to optimize bound-constraint optimization problems. The main aim of this work is to introduce a new metaheuristic called Spherical Search (SS). The main characteristics of the SS algorithm are as follows:

1. Less number of parameters.
2. Proper balance between exploration and exploitation.
3. Rotational-Invariance.
4. Maps the contour of search space.
5. Maintains high diversity during the optimization process.

The performance of the SS algorithm is assessed on two well-known benchmark suites viz. IEEE CEC 2014 and IEEE CEC 2011 problem suites. The obtained outcomes of proposed algorithm are compared with the state-of-art meta-heuristics. The comparative analysis proves that the SS algorithm performs better than the state-of-art meta-heuristics. Other sections of this paper are organized as follows:

- Section 2 discusses the proposed method and its operators.
- Section 3 discusses parameter adaptation approach to tune the parameters of SS online during the optimization process.
- Section 4 discusses experimental results and its comparison with other popular techniques.
- Section 5 lastly concludes the paper and discusses future work.

2. Spherical search optimization algorithm

In this section, the mathematical modeling of the SS algorithm is discussed and developed.

SS algorithm is a swarm-based meta-heuristic proposed to solve the non-linear bound-constrained global optimization problems. It shows some properties similar to other popular meta-heuristics, particularly PSO and DE. In the SS algorithm, the search space is represented in the form of vector space where the location of each individual in space is a position vector representing a candidate solution to the problem.

In a D-dimension search space, for each individual, $(D-1)$-spherical boundary is prepared towards the target direction in every iteration before generating the trial location of the individual. Here the target direction is the main axis of the spherical boundary and the individual lies on the surface of the spherical

Table 1
Metaheuristic optimization algorithms.

S.N.	Algorithm	Ref.
1.	Seagull Optimization Algorithm	[13]
2.	Whale Optimization Algorithm	[14]
3.	Ant Lion Optimizer	[15]
4.	Lion Optimization Algorithm	[16]
5.	Social Mimic Optimization Algorithm	[17]
6.	Thermal Exchange Optimization	[18]
7.	Snap-drift Cuckoo Search	[19]
8.	Virus Colony Search	[20]
9.	Grasshopper Optimisation Algorithm	[21]
10.	Salp Swarm Algorithm	[22]
11.	Squirrel Search Algorithm	[23]
12.	Interactive Search Algorithm	[24]
13.	Emperor Penguin Optimizer	[25]
14.	Coyote Optimization Algorithm	[26]
15.	Spotted Hyena Optimizer	[27]

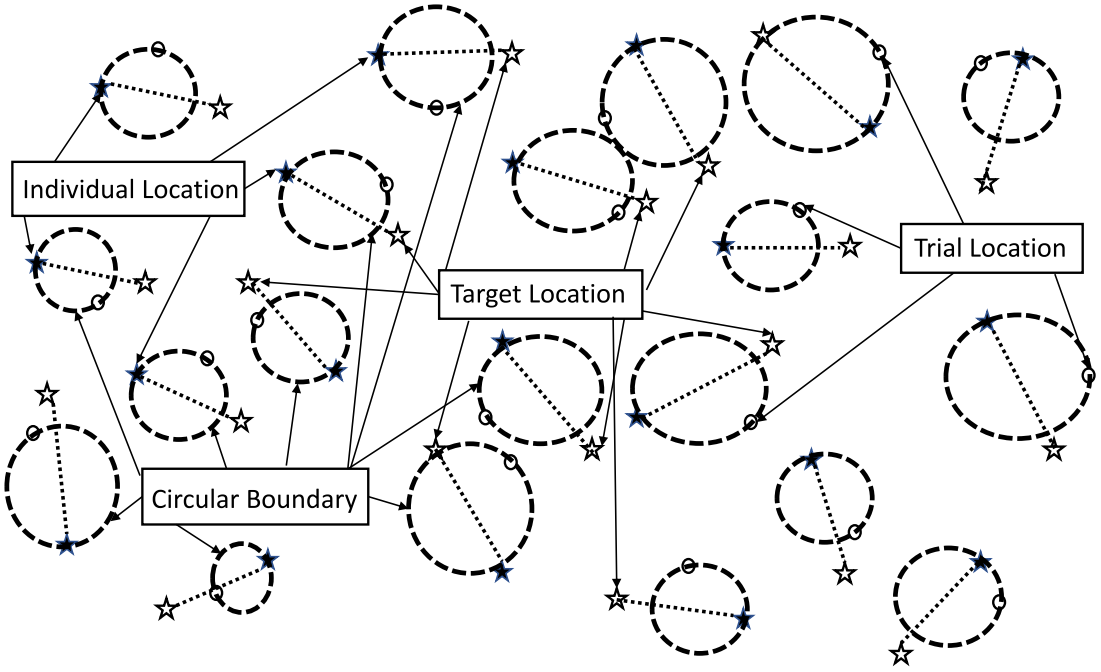

Fig. 1. Demonstrating the 1-spherical (circular) boundary of individuals of SS algorithm in 2-D search space.

boundary. An example of a 2-D search space is depicted in Fig. 1. In this Fig. 1-spherical boundary for each of individual is shown as –, target location has been shown by ☆ and each 1-spherical boundary is generated using axis obtained by individual location and target locations. Trial solutions appear on the 1-spherical boundary. Thus, in every iteration, the trial location for each individual is generated on the surface of $(D-1)$-spherical boundary. An objective function value determines the fitness value of a location. On the basis of the fitness value of the trial locations, better locations pass on into the next iteration as individual locations. In the SS algorithm, solution update procedure and spherical search movement balance the ability of exploration and exploitation. When the $(D-1)$-spherical boundary is small, exploitation of search-space is emphasized in the algorithm. On the other hand, in case of larger $(D-1)$-spherical boundary, exploration of the space gets emphasized. It is evident that when the target location of a respective individual is far-off the individual has a tendency to explore, as the spherical boundary is large. This is advantageous as in such conditions it is better to explore the larger search space. On the contrary, when the target locations of a respective individual are nearby, the individual has a tendency to exploit as the spherical boundary becomes small. This is advantageous as in such situations, it is better to exploit in a small search space.

At the end of every iteration, the location having the best fitness value is saved as the best solution. Stopping criteria is achieved when the number of function evaluations reaches to a specified number or when the value of the best solution reaches near to the predefined solution within a specified tolerance. For the reported experimental work of this paper, both the stopping criteria have been used. The main steps of SS algorithm depicted in the flowchart (Fig. 2) are described as follows:

2.1. Initialization of population

At the kth iteration, population P_x is represented as follows.

$$P_x^{(k)} = [\bar{x_1}^{(k)}, \bar{x_2}^{(k)}, \ldots \bar{x_N}^{(k)}] \tag{1}$$

where

$$\bar{x_i}^{(k)} = [x_{i1}^{(k)}, x_{i2}^{(k)}, \ldots x_{iD}^{(k)}]^T. \tag{2}$$

Here, x_{ij} is the value of jth element (parameter) of ith solution and D is the total number of elements (parameters). So, $\bar{x_i}$ is actually representing a point on D—dimensional search space. Here, x_{ij}^0 is initialized using random uniform distribution between

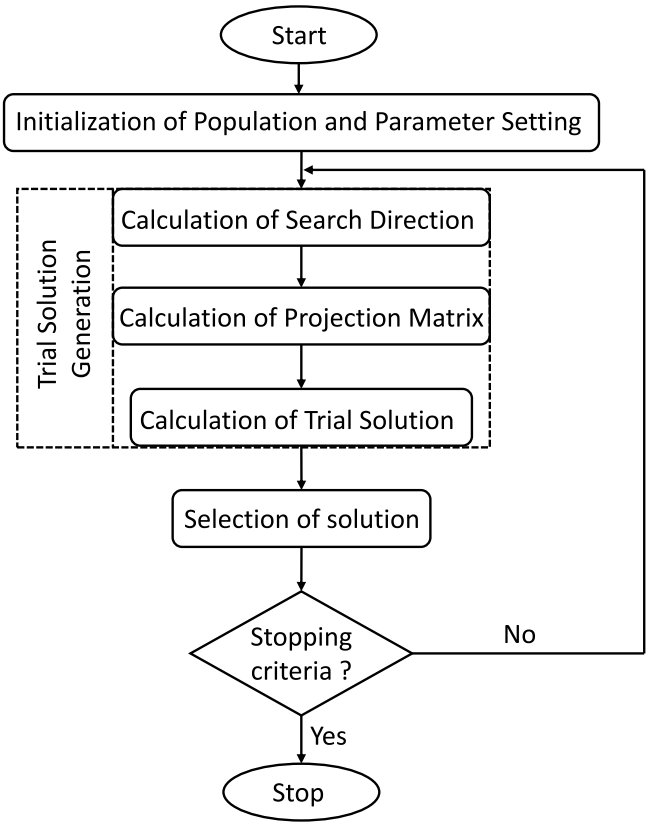

Fig. 2. Simple framework of SS algorithm.

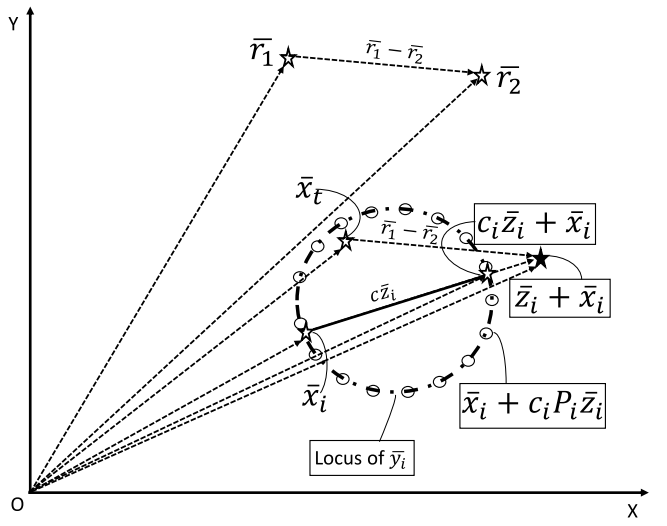

Fig. 3. Demonstrating the solution update scheme of SS algorithm in 2-D search space.

pre-specified lower and upper bounds of jth element as follows:

$$x_{ij}^0 = (x_{hj} - x_{lj}) * rand(0, 1] + x_{lj}, \tag{3}$$

where x_{hj} and x_{lj} represent the upper and lower bounds of jth element respectively. Also, $rand(0, 1]$ generates random number from uniform distribution within the limit $(0, 1]$.

2.2. Spherical surface and trial solutions

In the case of population-based optimization algorithms, in every iteration, there will be a need of calculation of new potential solutions that compete with the old solution to become a part of the population in the next iteration. In this algorithm, the term, trial solution, is used to represent these new potential solutions.

In SS algorithm, for each solution, a $(D-1)$-spherical boundary is prepared where the search direction passes through the main axis of boundary i.e search direction crosses the center of $(D-1)$-spherical boundary. A simple generation process of trial solutions, $\bar{y}_i$, is demonstrated on a 2-D search space in Fig. 3. It is seen from the Fig. 3, that the $(D-1)$-spherical boundary (locus of $\bar{y}_i$) is a circle (1-sphere) with diameter of $c_i\bar{z}_i$. Points represented by vectors $\bar{x}_i$ and $c_i\bar{z}_i + \bar{x}_i$ lie on the boundary of 1-sphere. Point represented by $\bar{z}_i$ is the search direction determined using vectors $\bar{r}_1$, $\bar{r}_2$ and $\bar{x}_t$. The determination of search direction $\bar{z}_i$ is described in Section-2.2.1

In SS algorithm, following equation is used to generate a trial solutions corresponding to i-th solution.

$$\bar{y}_i^{(k)} = \bar{x}_i^{(k)} + c_i^{(k)}P_i^{(k)}\bar{z}_i^{(k)}, \tag{4}$$

where, P_i is a projection matrix, which decides the value of $\bar{y}_i^k$ on the $(D-1)$-spherical boundary. For a particular solution, $x_i^{(k)}$, different possible values of $P_i^{(k)}$ yield different values of $\bar{y}_i^{(k)}$. Locus of $\bar{y}_i^{(k)}$ gives $(D-1)$-spherical boundary.

To define all the iterative steps of the SS algorithm, the calculation procedure of $\bar{z}_i^{(k)}$, $c_i^{(k)}$ and $P_i^{(k)}$ are discussed in following sections.

2.2.1. Calculation of search direction, $\bar{z}_i^{(k)}$

In optimization algorithms, quality of new solution highly depends upon the balance between exploration and exploitation of search space. Emphasis on exploration of search space increases the diversity of candidate solutions but slows the optimization process resulting in delayed or no convergence. Whereas, emphasis on exploitation may accelerate the optimization process which may lead to premature convergence trapped in local minima.

Search direction, $\bar{z}_i^{(k)}$, should be generated in such a way that it guides the i-th solution towards the better solutions. A simple illustration of calculation of search direction is shown in Fig. 3. From Fig. 3, it is clear that $\bar{x}_t$, r_1 and r_2 are needed to calculate the search direction as follows.

$$\bar{z}_i^{(k)} = (\bar{x}_t^{(k)} + r_1^{(k)} - r_2^{(k)}) - \bar{x}_i^{(k)}, \tag{5}$$

where, $\bar{x}_t$ is the target point. In Eq. (5), two random solutions r_1 and r_2 are selected from the current set of solutions (population). So, the actual search direction deviates by some angle from target direction, as shown in Fig. 3.

In this paper, two methods are introduced to calculate the search direction, namely *towards-rand* and *towards-best*. Method *towards-rand* has a better exploration capability and *towards-best* improves the exploitation capability. So to provide a good balance between the exploration and exploitation of search space, for the half population of better solution, calculation of search direction can be done by *towards-rand* and for the rest half of the population, *towards-best* is used to calculate the search direction thereby forcing diversity in the set of better solutions and forcing the inferior solutions to strive for improved fitness.

In *towards-rand*, the search direction, $\bar{z}_i^{(k)}$, for ith solution at kth iteration is calculated using following equation.

$$\bar{z}_i^{(k)} = \bar{x}_{p_i}^{(k)} + \bar{x}_{q_i}^{(k)} - \bar{x}_{r_i}^{(k)} - \bar{x}_i^{(k)}, \tag{6}$$

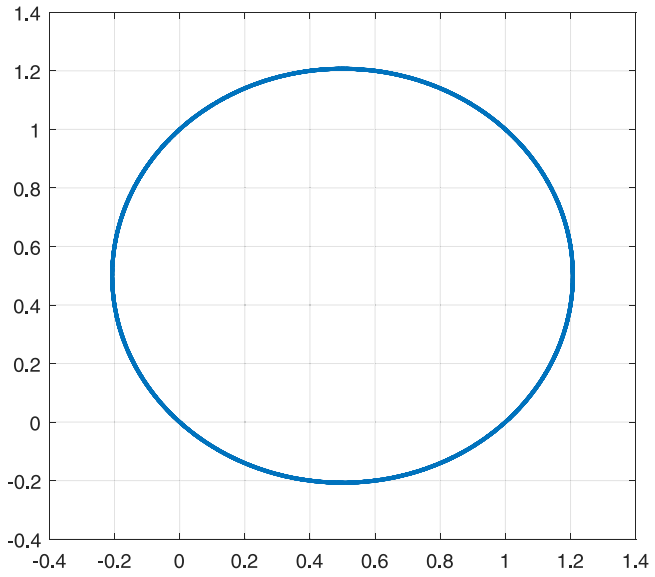

Fig. 4. Illustrating the locus of projection of point (1, 1).

where
p_i, q_i, and r_i are randomly selected indices from among 1 to N such that $p_i \neq q_i \neq r_i \neq i$.

While in *towards-best*, the search direction, $\bar{z}_i^{(k)}$, for ith solution at kth iteration is calculated using following equation.

$$\bar{z}_i^{(k)} = \bar{x}_{pbest_i}^{(k)} + \bar{x}_{q_i}^{(k)} - \bar{x}_{r_i}^{(k)} - \bar{x}_i^{(k)}, \tag{7}$$

where $x_{pbest_i}^{(k)}$ represents the randomly selected individual from among the top p solutions searched so far.

Here, $\bar{x}_{p_i}$ and $\bar{x}_{pbest_i}$ represent the target points in *towards-rand* and *towards-best* respectively. Difference term $(\bar{x_q} - \bar{x_r})$ is common in both *towards-rand* and *towards-best* which represents $\bar{r}_1 - \bar{r}_2$ in Eq. (5), which is an approximation of distribution of difference of solutions in a population. In calculation of new search direction, difference term $(\bar{x_q} - \bar{x_r})$ makes the population to evolve maintaining the diversity of solution thereby avoiding convergence to local minima.

2.2.2. Projection matrix

Projection matrix, P is a symmetrical matrix which is used for linear transformation from search space to itself such that $P^2 = P$ i.e. whenever P transforms a point twice, it provides the same point. Projection matrix, $P = A' diag(\bar{b}_i)A$, has been used in Eq. (4) to linearly transform $c_i\bar{z}_i + \bar{x}_i$ to generate trial solution $\bar{y}_i$ on the circular (1-spherical) boundary as shown in Fig. 3. Here, A and $\bar{b}_i$ are the orthogonal matrix and binary vector respectively. The total number of combinations of possible binary vectors are finite but in case of orthogonal matrix, A, the possible combinations are infinite. Therefore, all the possible projections of $c_i\bar{z}_i + \bar{x}_i$ create a $(D-1)$-spherical boundary on search space. To illustrate the locus of all possible projection of a point on a 2-D search space, 10,000 randomly generated samples of projection matrix, P, transforming a point (1, 1) are plotted in Fig. 4. It is seen from the Fig. 4 that locus of $P \times [1, 1]'$ is a circular ring passing through (0, 0) and (1, 1) with a diameter of $\sqrt{2}$ and center (0.5, 0.5).

Method to compute elements of P along with c has been illustrated as follows.

2.2.3. Orthogonal matrix, A

At the start of kth iteration, an orthogonal matrix, A, is generated randomly such that

$$AA' = I. \tag{8}$$

2.2.4. Binary diagonal matrix, diag($\bar{b}_i$)

Binary diagonal matrix, $diag(\bar{b}_i)$, are calculated randomly in such a way that,

$$0 < \text{rank}(diag(\bar{b}_i)) < D \tag{9}$$

2.2.5. Step-size control vector, $\bar{c}$

A step-size control vector, $\bar{c}^{(k)} = [c_1^{(k)}, c_2^{(k)}, \ldots . c_N^{(k)}]$, consists of step-size control parameter used for generation of all the possible trial-solutions where $c_i^{(k)}$ represents the step-size control parameter for ith trial-solution at kth iteration.

At the start of kth iteration, the elements of $\bar{c}^{(k)}$ are calculated randomly in range of [0.5 0.7], arrived by experiments.

2.3. Selection of new population for next iteration

Greedy selection procedure is applied to select new set of population for next iteration. To update the ith solution of the population, following criteria is applied. If the objective function value of trial solution, $f(\bar{y}_i^{(k)})$, is lower than the objective function value of solution, $f(\bar{x}_i^{(k)})$ then y_i replaces x_i.

Mathematically,

$$\bar{x}_i^{(k+1)} = \begin{cases} \bar{y}_i^{(k)}, & \text{if } f(\bar{y}_i^{(k)}) \leq f(\bar{x}_i^{(k)}) \\ \bar{x}_i^{(k)}, & \text{otherwise} \end{cases} \tag{10}$$

2.4. Stopping criteria

Termination of iterations depends upon two criteria: (i) the maximum number of function evaluations and (ii) convergence of solution i.e solution is not getting updated for specified number of consecutive iterations.

2.5. Steps of spherical search algorithm

The Pseudo-code of the proposed algorithm is shown in Algorithm 1 and the steps are summarized as follows:

- **Step 1:** Initialize the population P.
- **Step 2:** Calculate the objective function of each solution of P.
- **Step 3:** The best solution of population is selected as best solution.
- **Step 4:** Calculate the search direction for each solution of population P.
- **Step 5:** Calculate the orthogonal matrix, A.
- **Step 6:** Calculate parameters: c_i and *rank of projection matrix*.
- **Step 7:** Calculate trial solution for each solution of population P.
- **Step 8:** Update the population using greedy selection operator.
- **Step 9:** If the stopping criterion is satisfied then the algorithm will be stopped, otherwise it will return to **Step 3**.
- **Step 10:** Return the best optimal solutions, after stopping criteria is satisfied.

Algorithm 1 SS

procedure SPHERICAL SEARCH ALGORITHM
  Initialize the Population
  $c_i \leftarrow$ rand(0, 1]
  while $FEs < FE_{max}$ **do**
    $A \leftarrow$ **ComputeOrthogonalMatrix**()
    for $i = 1$ to N **do**
      $diag(\bar{b}_i) \leftarrow$ **ComputeBinaryVector**()
      if $i < 0.5 * N$ **then**
        $\bar{z}_i \leftarrow$ **TowardsRand**(i)/*Better Exploration */
      else
        $\bar{z}_i \leftarrow$ **TowardsBest**(i)/*Better Exploitation*/
      end if
      $\bar{y}_i \leftarrow \bar{x}_i + c_i A' diag(\bar{b}_i) A\bar{z}_i$
      $Of_i \leftarrow$ **ObjectiveFunction**($\bar{y}_i$)
      $FEs \leftarrow FEs + 1$
      $\bar{x}_i \leftarrow$ **Selection**($\bar{x}_i, \bar{y}_i$)
    end for
    $P1 \leftarrow$ Sort($P1$)
  end while
end procedure

2.6. Space and time complexity of SS algorithm

2.6.1. Space complexity

The space complexity of SS Algorithm is defined as the maximum amount of space required during the optimization process. Therefore, the space complexity of SS algorithm is $O(N \times D)$, where N is the size of population P and D is the dimension of solution space.

2.6.2. Time complexity

The time complexity of all the steps of SS algorithm are as follows:

1. Initialization of population in SS algorithm requires $O(N \times D)$ time.
2. Calculation of objective function of each solution requires $O(FE_{max} \times D) = O(Max_{iter} \times N \times D)$ time during the optimization process, where FE_{max} is the maximum number of allowed function evaluation.
3. Calculation of orthogonal matrix requires $O(Max_{iter} \times D \times \log(D))$ time during the optimization process.
4. Calculation of trial solutions requires $O(Max_{iter} \times N)$ time.

Hence, the overall time complexity of proposed algorithm is $O(Max_{iter} \times N \times D \times \log(D)) = O(FE_{max} \times D \times \log(D))$.

2.7. SS algorithm vs. PSO and DE

In this subsection, to see the effect of the proposed SS algorithm with respect to PSO, and DE, Griewank function [29] is simulated as a benchmark problem. For minimization of Griewank function, all algorithms are employed using 100 individuals for 200 iterations.

To describe the performance of SS algorithm with respect to PSO and DE, four qualitative metrics: **Search History**, **Search Trajectory of individual**, **Population diversity**, and **Convergence curve** are used to describe the performance of SS algorithm.

2.7.1. Search history

To indicate the performance of algorithms in terms of their exploitation and exploration capability, distribution of solutions onto the search space for iteration = 40, 80, 120, 160, and 200; is demonstrated in Fig. 5. Apparently, the high distribution density symbolizes the exploitation of the local region of solution-space and the low distribution density means the exploration of the global region of solution-space.

It is obvious that the individuals are highly distributed in initial iterations and converge to the optimal solution in final iterations. But, during the optimization process, the diversity should be maintained to search better location of search-space for better solutions. From Fig. 5, it is seen that the SS algorithm maintains better diversity as compared to PSO and DE. In the SS algorithm, the distribution of solutions is sparse in the worst region of search space and is dense in the region closer to all local and global optimum.

2.7.2. Search trajectory

The Search trajectory of an individual is one of the important metrics that can completely describe the exploitation and exploration of the algorithm. Fig. 6 represents the trajectory of randomly selected individuals when the problem is being solved using the SS, PSO, and DE algorithm. In the SS algorithm, the trajectory of selected individuals shows large-scale variations in the early iterations as compared to PSO and DE. With the increase of iterations, such variations reduce, and the positions of individuals mature uniformly and continuously tend to secure the location at global optimum without falling in local optimum in the later iterations. Consequently, solutions of PSO and DE fails to secure global optimum position and slips to a local optimum. Apparently, the large variations in the earlier iterations symbolize the exploration property, and the small variations in the following iterations show the exploitative search for the better region of search-space. Exploitation and exploration are cooperative during the optimization process that is presented in the search histories and trajectory of a solution.

2.7.3. Population diversity

Furthermore, it can be observed from the population diversity curve of the algorithms shown in Fig. 7 that SS algorithm presents better diversity than the PSO, and DE during the optimization process. Therefore, the population diversity curve together with the search histories and trajectory of an individual exhibit that SS algorithm attains a better balance between the exploration and exploitation properties as compared to PSO and DE.

2.7.4. Convergence curve

In the final stage of the optimization process, the algorithm should obtain the global optimum precisely and rapidly. But above-discussed metrics cannot address it. The convergence curve is the most popular qualitative metric in estimating the convergence performance of the algorithms. The Fig. 8 shows the convergence curves which represents the convergence rate of SS, PSO, and DE for 2-D and 20-D search spaces. As shown in Fig. 8, the convergence curve of 2-D is very smooth and drops rapidly in the case of PSO and DE, illustrating the larger contribution of exploitation in PSO and DE than the exploration as compared to SS algorithm. However, exploration is dominant in the SS algorithm. In contradiction, for 20-D search space, convergence curves of PSO and DE are very rough and drop slowly, which means that the exploration operators are dominant in PSO and DE as compared to SS algorithm. Finally, in the case of PSO and DE, their convergence curves stagnate and cannot approximate the global optimum in the final iterations. On the other hand, for the SS algorithm, the convergence performance and the accuracy of the final approximation to the global optimal solution is better than the PSO, and DE.

Based on the above discussion, the SS algorithm establishes a better balance among the exploitation and exploration and search performance for the global optimum which is also satisfactory as compared to PSO and DE.

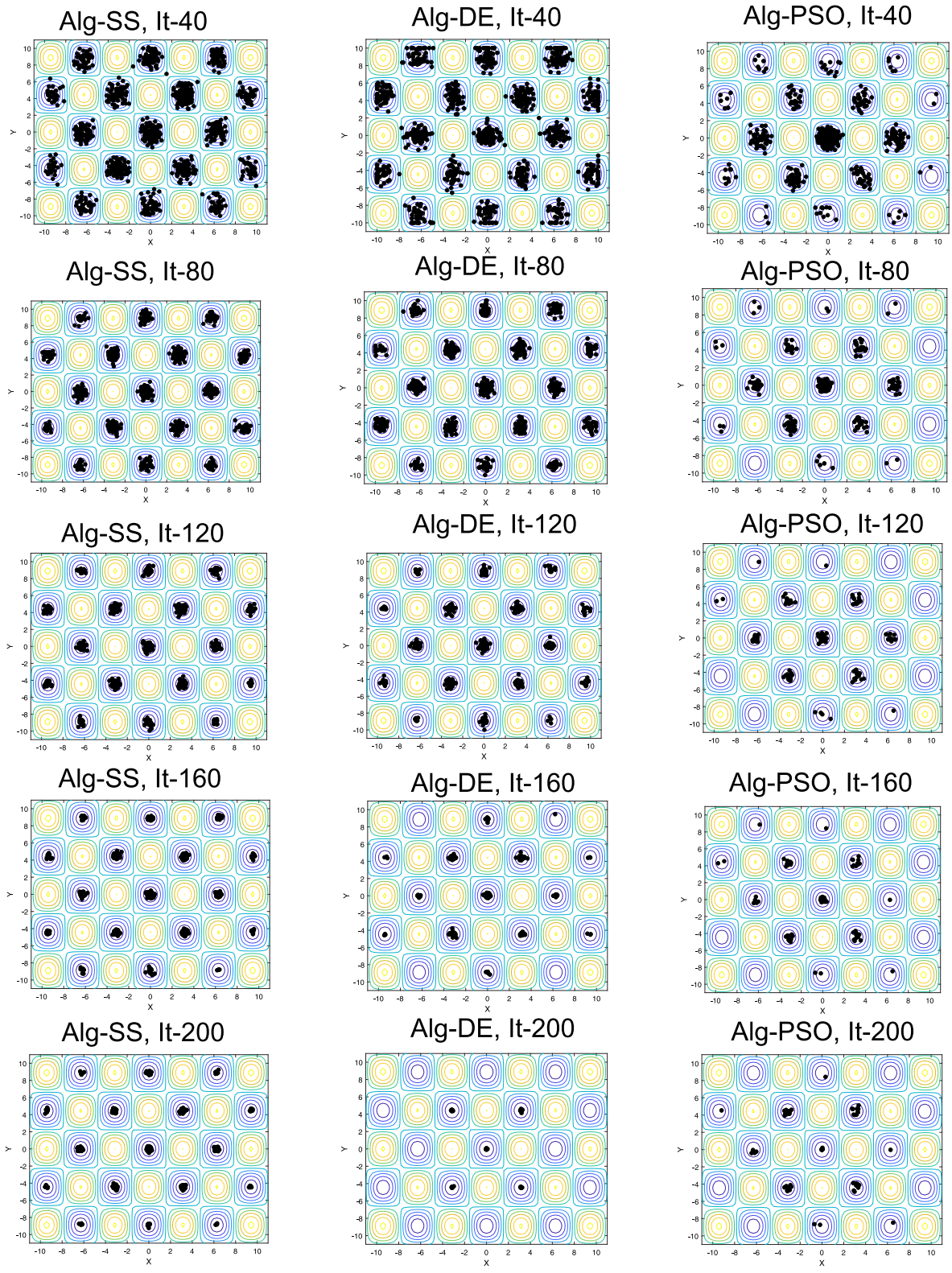

Fig. 5. Search-History of individuals of SS, DE, and PSO with respect to iteration.

3. Parameter adaptation

The performance of SS is highly dependent on the control parameters c_i and *rank* and the size of population N. Therefore, when SS is applied to the real-life optimization problems, there is the need for fine-tuning of these parameters in order to obtain the better solution with fast convergence rate. Since this is a common practice, an online self-adaptive approach is considered as a better option to set the parameter values automatically during the search.

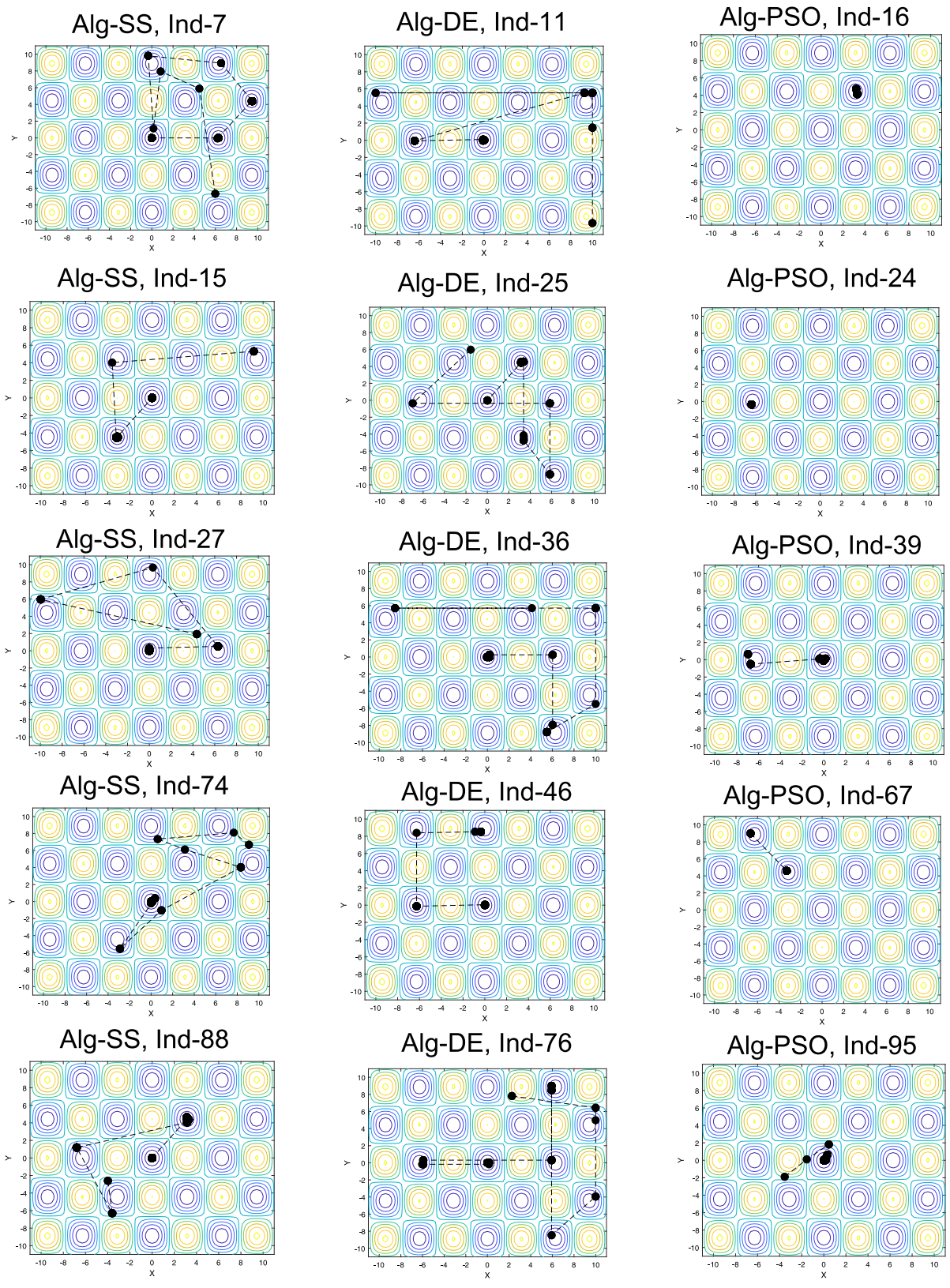

Fig. 6. Trajectory of randomly selected individuals of SS, DE, and PSO.

3.1. Exponential population size reduction

In order to improve the performance of SS, we incorporate a population size reduction method for dynamically resizing the population during an iteration. We use Exponential Population Size Reduction (EPSR) which reduces the population exponentially as a function of the number of iteration as proposed in [30]. EPSR continuously reduces the populations to match a exponential function where the population size at iteration 1 are N^{init} and the population size at the end of the run are N^{min}. After each

iteration k, the population size for the next iteration, $N^{(k+1)}$, is computed according to the formula:

$$N^{(k+1)} = round\left(N^{init}\left(1 - \frac{N^{init} - N^{min}}{nfes_{max}}\right)\right)^{k}. \quad (11)$$

N^{min} is set to the smallest possible value such that the evolutionary operators can be applied. In the case of SS, $N^{min} = 4$ because the *TowardsRand* operator requires 4 individuals. $nfes_{max}$ is the allowed maximum number of function evaluations. Whenever $N^{(k+1)} < N^{(k)}$, the $(N^{(k)} - N^{(k+1)})$ worst-ranking individuals are deleted from the populations.

3.2. Success history based control parameter adaptation

Success history based different strategies have been studied and proposed by the many researchers for adapting the control parameters, selection of solution update strategy from the pool, and selection of an algorithm for next run in case of hybrid algorithms. In success history based strategies, algorithms use the past history of their success i.e. learn from the previous iterations to adapt the control parameters or select a strategy from the selection pool for the current iteration. In this paper, a similar idea, Success History-based-control Parameter Adaptation (SHPA) procedure, is proposed to adapt the two control parameter, *rank* and c_i during the search.

The framework of SHPA is similar to the history based parameter adaptation proposed in [31]. In [31], a history based parameter adaptation is proposed to adapt the crossover probability and scaling factor during the search, where it creates a historical matrix, L, of size $(2 \times H)$ to save the H entries for both control parameters [31]. Similarly, in SHPA, a historical matrix of size $(2 \times H)$ is maintained to save the learning values l_r and l_c for parameters *rank* and c, respectively, for the last H iterations. Further this matrix L is used to generate the control parameters in calculation of all trial solutions for the next iteration. Generation procedure of $rank_i$ and c_i are discussed in following sections.

3.2.1. Generation of $rank_i^{(k)}$

At each iteration k, in the calculation of ith trial-solution, the rank of projection matrix, $rank_i^{(k)}$, is independently generated from binomial distribution with number of trials D and probability L_{1j}.

$$rank_i^{(k)} = Binornd(D, L_{1j}) \quad (12)$$

where *Binornd* represents the binomial distribution and j is randomly chosen from column of matrix L for every i independently. Then, $rank_i^{(k)}$ is truncated to $[1, D]$.

3.2.2. Generation of $c_i^{(k)}$

At each iteration k, a step-size control factor, c_i, is generated from the Cauchy distribution of mean L_{2j} and standard deviation 0.1 as shown in Eq. (13).

$$c_i = Cauchyrand(L_{2,j}, 0.1) \quad (13)$$

where *Cauchyrand* represents the Cauchy distribution and j is randomly chosen from column of matrix L for every i independently. Then, c_i is truncated to [0, 1].

3.2.3. Calculation of l_r and l_c

At the end of each iteration, calculation of the learning values, l_r and l_c, is required. For this purpose, two vectors S_r and S_c, containing the *rank* and c of successful trials respectively, are created. Then, the l_r and l_c are calculated using following equations.

$$l_r^{(k)} = \frac{\sum_{h=1}^{|S_r^{(k)}|} w_h^{(k)} r_h^{(k)2}}{\sum_{h=1}^{|S_r^{(k)}|} w_h^{(k)} r_h^{(k)}} \quad (14)$$

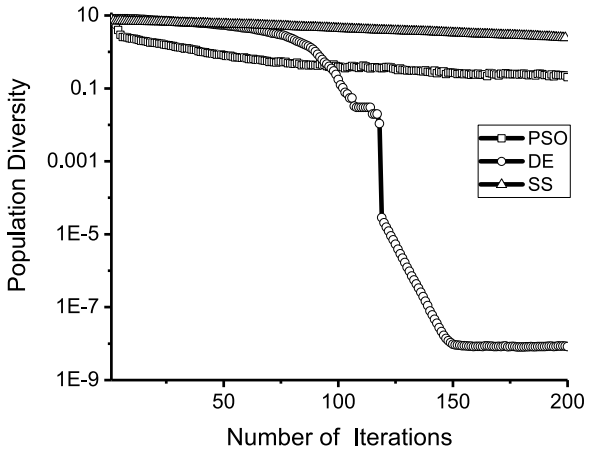

Fig. 7. Population diversity of SS, DE, and PSO with respect to iteration.

$$l_c^{(k)} = \frac{\sum_{h=1}^{|S_c^{(k)}|} w_h^{(k)} c_h^{(k)2}}{\sum_{h=1}^{|S_c^{(k)}|} w_h^{(k)} c_h^{(k)}} \quad (15)$$

where $|S_r^{(k)}|$ and $|S_c^{(k)}|$ are the length of vectors $S_r^{(k)}$ and $S_c^{(k)}$ respectively. $w_h^{(k)}$ is calculated using following equation.

$$w_h^{(k)} = \frac{f_h^{(k)} - f_h^{(k-1)}}{\sum_{g=1}^{|S_r^{(k)}|}(f_g^{(k)} - f_g^{(k-1)})} \quad (16)$$

4. Experimental results and discussion

In this section, the performance of SS algorithm is evaluated on the bound-constrained optimization problems. The well-known 30 benchmark problems from special session of IEEE CEC 2014 [32] are considered to analyze the performance of SS algorithm as compared with the other state-of-art algorithms. Detailed information and characteristics of these problems are available in [32]. To assess the performance of SS algorithm, four experiments are designed which are as follows.

1. Experiment 1: In first experiment, parameter sensitivity analysis is done.
2. Experiment 2: In second experiment, the effectiveness of different components of SS are studied in terms of exploitation, exploration and population diversity.
3. Experiment 3: In third experiment, the performance of basic SS (without self-adaptation of parameters) is compared with the state-of-art algorithms.
4. Experiment 4: Finally, the performance of SASS (with self-adaptation of parameters) is compared with the best performer of the IEEE CEC competitions.

All the experiments in this section are conducted on a personal computer with Intel(R) Core(TM) i5-3470 3.20 GHz processor and 10 GB of RAM using MATLAB 2015b. The source code of SS used in experiments can be downloaded from this link: https://github.com/abhisheka456/spherical-search.

4.1. Experiment 1: Parameter sensitivity analysis

The sensitivity of all parameters of SS is experimentally investigated. A set of 30 test problems with 30D from IEEE CEC 2014 benchmark suite is utilized as the benchmark problems to study the sensitivity of parameters of SS. Additionally, Wilcoxon's

Table 2
Mean and SD of best error value obtained in 51 independent runs by SS, SStb, SStr, and SSr on 30-*D* CEC2014 problem suite (Mean: Mean of best error, SD: Standard deviation of best error, W: result of Wilcoxon signed rank test).

Prob	SS		SStb			SStr			SSr		
	Mean	SD	Mean	SD	W	Mean	SD	W	Mean	SD	W
1	8.75E+03	6.70E+03	**6.19E+03**	**6.69E+03**	−	1.73E+05	1.23E+05	+	7.25E+03	5.99E+03	=
2	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	=	8.41E+03	5.76E+03	+	**0.00E+00**	**0.00E+00**	=
3	1.03E−07	2.81E−07	**1.00E−09**	**4.29E−09**	=	5.16E+01	3.57E+01	+	3.96E−07	8.56E−07	=
4	**0.00E+00**	**0.00E+00**	2.49E+00	1.24E+01	=	6.40E+01	2.78E+01	+	1.24E+00	8.88E+00	+
5	**2.09E+01**	**5.88E−02**	2.09E+01	5.91E−02	=	2.10E+01	5.81E−02	+	2.10E+01	4.32E−02	=
6	**2.79E−01**	**5.86E−01**	9.41E−01	2.67E+00	+	4.14E+00	5.04E+00	+	5.77E−01	1.90E+00	+
7	**0.00E+00**	**0.00E+00**	1.11E−03	3.17E−03	+	9.51E−03	9.89E−03	+	6.28E−04	2.20E−03	+
8	1.59E+02	1.00E+01	1.63E+02	1.13E+01	=	**1.58E+02**	**9.05E+00**	=	1.60E+02	8.54E+00	=
9	1.61E+02	1.04E+01	1.68E+02	9.19E+00	+	**1.59E+02**	**1.04E+01**	=	1.62E+02	9.55E+00	=
10	**6.32E+03**	**2.55E+02**	6.33E+03	2.80E+02	=	6.48E+03	2.71E+02	+	6.43E+03	3.21E+02	+
11	**6.57E+03**	**3.04E+02**	6.67E+03	3.11E+02	=	6.82E+03	3.26E+02	+	6.77E+03	2.46E+02	+
12	**2.29E+00**	**3.10E−01**	2.50E+00	2.52E−01	+	2.43E+00	2.77E−01	+	2.54E+00	2.74E−01	+
13	2.49E−01	3.21E−02	2.44E−01	4.98E−02	=	2.75E−01	3.69E−02	+	**2.39E−01**	**3.21E−02**	=
14	2.51E−01	3.53E−02	**2.47E−01**	**4.95E−02**	=	2.68E−01	2.84E−02	+	2.65E−01	3.61E−02	+
15	**1.39E+01**	**7.80E−01**	1.43E+01	1.08E+00	+	1.39E+01	8.85E−01	=	1.39E+01	1.05E+00	=
16	**1.19E+01**	**3.28E−01**	1.23E+01	2.03E−01	+	1.25E+01	2.51E−01	+	1.24E+01	2.20E−01	+
17	**4.80E+02**	**2.34E+02**	6.52E+02	3.70E+02	+	1.12E+03	3.17E+02	+	9.58E+02	3.41E+02	+
18	7.33E+01	2.28E+01	7.72E+01	2.38E+01	=	6.76E+01	1.77E+01	=	**5.42E+01**	**1.09E+01**	−
19	5.37E+00	6.72E−01	5.17E+00	6.17E−01	=	5.16E+00	5.39E−01	=	**5.02E+00**	**5.42E−01**	−
20	5.77E+01	2.14E+01	4.13E+01	5.73E+00	−	4.39E+01	9.88E+00	−	**3.51E+01**	**5.14E+00**	−
21	**4.85E+02**	**1.96E+02**	6.36E+02	2.49E+02	+	7.86E+02	1.91E+02	+	7.02E+02	2.15E+02	+
22	**2.29E+02**	**1.06E+02**	3.31E+02	1.29E+02	+	4.15E+02	9.14E+01	+	3.64E+02	1.15E+02	+
23	**3.15E+02**	**5.64E−13**	3.15E+02	5.19E−13	+	3.15E+02	1.18E−04	+	3.15E+02	5.16E−13	+
24	**2.21E+02**	**7.79E+00**	2.24E+02	8.54E+00	+	2.27E+02	4.19E+00	+	2.24E+02	4.76E+00	+
25	**2.03E+02**	**2.74E−01**	2.03E+02	1.44E−01	=	2.03E+02	3.85E−01	+	2.03E+02	2.06E−01	=
26	**1.00E+02**	**3.24E−02**	1.00E+02	3.98E−02	=	1.00E+02	3.55E−02	+	1.00E+02	3.41E−02	+
27	3.34E+02	4.34E+01	3.39E+02	4.19E+01	=	3.42E+02	2.63E+01	=	**3.11E+02**	**2.13E+01**	−
28	8.47E+02	9.35E+01	**8.43E+02**	**4.58E+01**	=	8.80E+02	4.33E+01	+	8.50E+02	4.37E+01	=
29	8.24E+02	6.91E+01	**8.11E+02**	**6.51E+01**	=	2.85E+03	4.00E+03	+	8.51E+02	2.74E+02	=
30	1.74E+03	8.95E+02	1.82E+03	7.31E+02	=	**1.27E+03**	**6.33E+02**	−	1.33E+03	5.64E+02	−
	+/=/−		11/17/2			22/6/2			14/11/5		

Table 3
Experimental results of SS with varying c on 30 test problems with 30D from IEEE CEC 2014 benchmark suite.

c	0.2	0.4	0.5	0.6	0.8	1.0
+	21	19	/	6	7	16
−	3	2	/	3	5	7
=	6	9	/	21	18	7

Table 4
Experimental results of SS with varying *rank* on 30 test problems with 30D from IEEE CEC 2014 benchmark suite.

rank	0.1D	0.3D	0.5D	0.7D	0.9D
+	18	15	/	14	16
−	2	5	/	5	7
=	10	10	/	11	7

Table 5
Experimental results of SS with varying p on 30 test problems with 30D from IEEE CEC 2014 benchmark suite.

p	0	0.05	0.1	0.15	0.2
+	20	18	/	17	22
−	3	4	/	3	5
=	7	8	/	10	3

signed rank test (W) at the 0.05 significance level is employed to compare the performance of algorithms.

The sensitivities of three important parameters such as c, *rank*, and p of SS algorithm are studied. In the case of c, six different values of c in SS, i.e. $c = 0.2$, $c = 0.4$, $c = 0.5$, $c = 0.6$, $c = 0.8$, and $c = 1.0$ are considered in this analysis. The outcomes of WT with respect to the SS with $c = 0.5$ is reported in Table 3. According to the WT's outcomes, SS with $c = 0.5$ performs better than SS with $c = 0.2$, $c = 0.4$, $c = 0.6$, $c = 0.7$, and $c = 0.9$ on 21, 19, six, seven, and 16 test problems, respectively. However, SS with $c = 0.5$ is outperformed by SS with $c = 0.2$, $c = 0.4$, $c = 0.6$, $c = 0.8$, and $c = 1.0$ on three, two, three, five, and seven test problems, respectively. From this analysis, it can be concluded that the performance of SS is highly sensitive to c and $c = (0.5, 0.7)$ is recommended in this work.

For *rank*, five different values are considered, i.e. $rank = 0.1D$, $rank = 0.2D$, $rank = 0.3D$, $rank = 0.5D$, $rank = 0.7D$, and $rank = 0.9D$ to analyze the sensitivity of *rank* in the performance of the SS. The results of WT with respect to the SS with $rank = 0.5D$ are reported in Table 4. It is seen from Table 4 that the performance

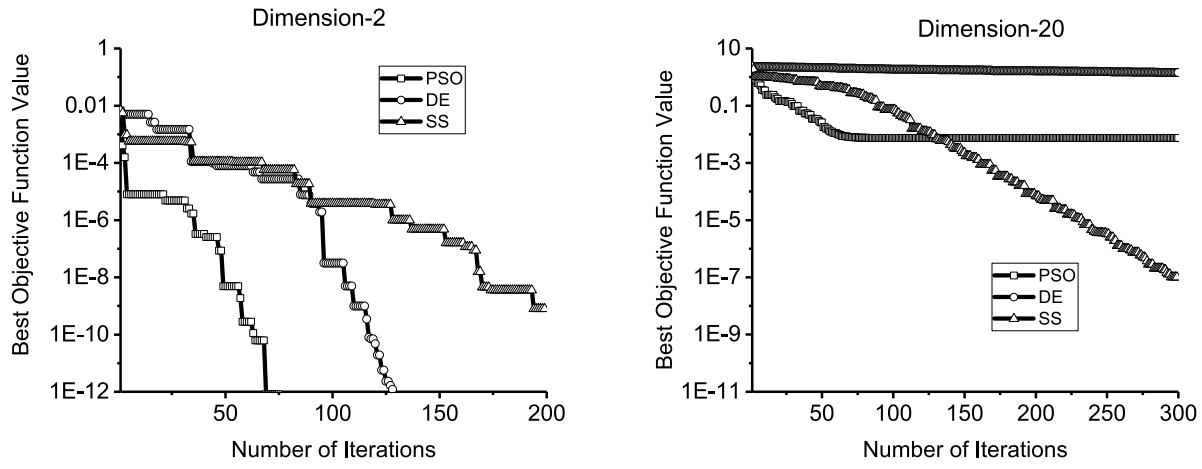

Fig. 8. Convergence of solutions of SS, DE, and PSO with respect to iterations.

Table 6
Mean and SD of best error value obtained in 51 independent runs by SS, PSO, BB-PSO, CLPSO, APSO, and OLPSO on 30-*D* CEC2014 problem suite (Mean: Mean of best error, SD: Standard deviation of best error, W: result of Wilcoxon signed rank test).

Nos	SS		PSO			BB-PSO			CLPSO			APSO			OLPSO		
	Mean	SD	Mean	SD	W	Mean	SD	W	Mean	SD	W	Mean	SD	W	Mean	SD	W
f01	**8.75E+03**	**6.70E+03**	5.14E+07	2.24E+07	+	2.40E+06	1.97E+06	+	7.82E+06	2.44E+06	+	2.69E+09	3.28E+08	+	6.12E+06	3.58E+06	+
f02	**0.00E+00**	**0.00E+00**	2.42E+08	4.23E+08	+	1.84E+03	7.68E+03	+	8.93E+01	2.33E+02	+	1.02E+11	2.29E+09	+	1.28E+03	1.48E+03	+
f03	**1.03E−07**	**2.81E−07**	5.97E+04	1.07E+04	+	6.23E+03	1.40E+04	+	2.03E+02	1.85E+02	+	1.19E+06	1.25E+06	+	3.23E+02	5.69E+02	+
f04	**0.00E+00**	**0.00E+00**	1.66E+02	8.29E+01	+	5.99E+01	3.45E+01	+	6.92E+01	2.06E+01	+	2.49E+04	1.54E+03	+	8.64E+01	2.22E+01	+
f05	2.09E+01	5.88E−02	2.08E+01	9.01E+01	=	**2.01E+01**	**5.20E−02**	−	2.04E+01	4.13E−02	−	2.13E+01	5.61E−02	+	2.03E+01	1.28E−01	−
f06	**2.79E−01**	**5.86E−01**	2.58E+01	6.40E+01	+	1.17E+01	2.12E+00	+	1.23E+01	1.47E+00	+	4.80E+01	1.79E+00	+	5.09E+00	1.48E+00	+
f07	**0.00E+00**	**0.00E+00**	9.28E+01	1.23E+01	+	2.73E−02	2.71E−02	+	3.43E−06	4.28E−06	+	1.06E+03	3.85E+01	+	1.02E−13	3.47E−14	=
f08	1.59E+02	1.00E+01	5.09E−02	1.11E+00	−	1.02E−03	8.96E−04	−	**0.00E+00**	**0.00E+00**	−	5.03E+02	3.02E+01	+	**0.00E+00**	**0.00E+00**	−
f09	1.61E+02	1.04E+01	3.18E+01	1.28E+02	−	**3.16E+01**	**8.10E+00**	−	5.22E+01	6.38E+00	−	4.78E+02	6.30E+00	+	4.06E+01	7.02E+00	−
f10	6.32E+03	2.55E+02	7.61E+02	7.61E+02	−	1.59E+00	1.42E+00	−	3.11E+00	1.60E+00	−	9.30E+03	5.68E+02	+	**8.72E−02**	**2.04E−01**	−
f11	6.57E+03	3.04E+02	6.83E+03	1.93E+03	+	**2.11E+03**	**4.38E+02**	−	2.33E+03	3.19E+02	−	9.24E+03	4.86E+02	+	2.28E+03	4.66E+02	−
f12	2.29E+00	3.10E−01	2.86E+00	2.77E+01	+	**1.66E−01**	**5.61E−02**	−	3.94E−01	5.53E−02	−	5.91E+00	1.32E+00	+	2.28E−01	6.38E−02	−
f13	2.49E−01	3.21E−02	6.40E−01	4.44E−01	+	**2.38E−01**	**5.24E−02**	=	3.17E−01	3.75E−02	+	1.03E+01	7.53E−01	+	2.59E−01	3.20E−02	+
f14	2.51E−01	3.53E−02	4.21E−01	1.57E−01	+	5.75E−01	2.78E−01	+	2.51E−01	2.85E−02	=	3.95E+02	2.22E+01	+	**2.41E−01**	**2.66E−02**	=
f15	1.39E+01	7.80E−01	**4.47E+00**	**1.92E+00**	−	7.82E+00	3.03E+00	−	7.49E+00	1.02E+00	−	1.05E+06	0.00E+00	+	6.67E+00	1.62E+00	−
f16	1.19E+01	3.28E−01	1.29E+01	4.82E−01	+	**9.13E+00**	**1.08E+00**	−	1.04E+01	3.72E−01	−	1.42E+01	2.37E−01	+	1.17E+01	5.48E−01	=
f17	**4.80E+02**	**2.34E+02**	3.26E+05	2.57E+05	+	6.44E+05	6.17E+05	+	9.59E+05	4.41E+05	+	2.86E+08	1.28E+08	+	7.98E+05	4.13E+05	+
f18	**7.33E+01**	**2.28E+01**	1.02E+06	2.95E+06	+	6.46E+03	8.74E+03	+	1.01E+02	4.67E+01	+	8.75E+09	3.11E+09	+	3.58E+02	5.12E+02	+
f19	**5.37E+00**	**6.72E−01**	2.82E+02	1.98E+01	+	7.43E+00	1.29E+00	+	7.39E+00	6.27E−01	+	8.45E+02	1.15E+02	+	6.13E+00	8.20E−01	+
f20	**5.77E+01**	**2.14E+01**	1.08E+04	2.18E+03	+	3.02E+03	2.91E+03	+	3.14E+03	1.40E+03	+	1.59E+07	1.37E+07	+	5.58E+03	4.01E+03	+
f21	**4.85E+02**	**1.96E+02**	7.54E+05	4.21E+05	+	2.28E+05	3.29E+05	+	8.46E+04	5.35E+04	+	1.33E+08	7.50E+07	+	1.07E+05	8.33E+04	+
f22	2.29E+02	1.06E+02	8.24E+02	5.24E+02	+	2.43E+02	2.10E+02	+	2.08E+02	7.41E+01	+	1.31E+04	9.38E+03	+	**2.20E+02**	**1.07E+02**	=
f23	3.15E+02	6.61E−03	3.42E+02	6.24E+00	+	3.15E+02	1.87E−12	+	3.15E+02	2.02E−06	+	**2.00E+02**	**0.00E+00**	−	3.15E+02	1.23E−10	+
f24	2.21E+02	7.79E+00	2.05E+02	2.41E−01	−	2.28E+02	4.97E+00	+	2.25E+02	4.53E−01	+	**2.00E+02**	**0.00E+00**	−	2.24E+02	5.47E−01	+
f25	2.03E+02	2.74E−01	2.20E+02	4.51E+00	+	2.06E+02	2.27E+00	+	2.08E+02	1.06E+00	+	**2.00E+02**	**0.00E+00**	−	2.09E+02	1.75E+00	+
f26	1.00E+02	3.24E−02	1.00E+02	2.42E−01	=	1.00E+02	5.49E−02	+	1.00E+02	6.42E−02	+	1.86E+02	2.68E+01	+	1.00E+02	4.44E−02	+
f27	3.34E+02	4.34E+01	2.51E+03	4.82E+02	+	6.53E+02	5.06E+01	+	4.14E+02	4.92E+00	=	**2.00E+02**	**0.00E+00**	−	3.26E+02	3.80E+01	=
f28	8.47E+02	9.35E+01	1.81E+03	4.91E+02	+	9.10E+02	2.59E+01	+	9.08E+02	3.97E+01	+	**2.00E+02**	**0.00E+00**	−	8.73E+02	2.97E+01	+
f29	8.24E+02	6.91E+01	8.77E+07	3.24E+07	+	1.70E+06	3.47E+06	+	9.71E+02	9.14E+01	+	**2.00E+02**	**0.00E+00**	−	1.36E+03	2.82E+02	+
f30	1.74E+03	8.95E+02	4.11E+05	1.87E+04	+	7.94E+03	1.07E+03	+	3.39E+03	7.88E+02	+	**2.00E+02**	**0.00E+00**	−	2.39E+03	5.99E+02	+
+/=/−			23/2/5			21/1/8			20/2/8			23/0/7			18/5/7		

of SS with $rank = 0.5D$ is better than SS with $rank = 0.1D$, $rank = 0.3D$, $rank = 0.7D$, and $rank = 0.9D$ on 18, 15, 14, and 16 test problems out of 30 test problems, respectively. Although, the performance of SS with $rank = 0.5D$ is not better than SS with $rank = 0.1D$, $rank = 0.3D$, $rank = 0.7D$, and $rank = 0.9D$ on 10, 10, 11, and seven test problems, respectively. On the basis of the outcomes of WT, it can be concluded that the performance of SS is highly sensitive to *rank*. The value of *rank* is set at $0.5D$ in SS.

Similarly, five different variants of SS with different values of p such as 0, 0.05, 0.1, 0.15, and 0.2 are utilized to analyze the sensitivity of p on the performance of SS. The results of this analysis in terms of WT with respect to the SS with $p = 0.1$ are reported in Table 5. As reported in Table 5, SS with $p = 0.1$ outperforms the SS with $p = 0$, $p = 0.05$, $p = 0.15$, and $p = 0.2$ on 20, 18, 17, and 22 test problems, respectively. However, SS with $p = 0.1$ is outperformed by SS with $p = 0$, $p = 0.05$, $p = 0.15$, and $p = 0.2$ on seven, eight, ten and three test problems, respectively. From the outcomes of WT, we can conclude that the performance of SS is very sensitive to the value of p. This analysis suggests the $p = 0.1$ in this work.

4.2. Experiment 2: Effectiveness of components of SS

The SS algorithm has utilized two types of search direction calculation operators: *towards-rand*, and *towards-best*, to balance the exploitative and exploratory search during the optimization process. Here, *towards-rand* provides a better exploration characteristic and *towards-best* shows a better exploitation characteristic. In order to implement a good balance between the

Table 7
Mean and SD of best error value obtained in 51 independent runs by SS, CoBiDE, FCDE, RSDE, POBL_ADE, and DE_best on 30-*D* CEC2014 problem suite (Mean: Mean of best error, SD: Standard deviation of best error, W: result of Wilcoxon signed rank test).

Nos	SS		CoBiDE			FCDE			RSDE			POBL_ADE			DE_best		
	Mean	SD	Mean	SD	W	Mean	SD	W	Mean	SD	W	Mean	SD	W	Mean	SD	W
f01	8.75E+03	6.70E+03	**3.24E+00**	**3.24E+00**	−	1.05E+05	9.10E+04	+	1.50E+03	1.70E+03	−	1.60E+04	1.22E+04	+	2.46E+07	9.27E+06	+
f02	**0.00E+00**	**0.00E+00**	1.12E−02	5.54E−03	+	**0.00E+00**	**0.00E+00**	=	**1.19E−09**	**5.99E−09**	=	3.14E+02	7.52E+02	+	**0.00E+00**	**0.00E+00**	=
f03	**1.03E−07**	**2.81E−07**	5.17E−07	4.32E−07	=	1.09E+02	4.04E+02	+	4.74E−02	1.16E−01	+	**6.43E−10**	**4.59E−09**	=	3.32E−05	2.00E−05	−
f04	**0.00E+00**	**0.00E+00**	1.85E+01	6.45E−01	+	2.99E+01	3.05E+01	+	3.05E+00	1.34E+01	+	6.34E+01	2.63E+01	+	7.68E+01	2.42E+01	+
f05	2.09E+01	5.88E−02	2.06E+01	5.12E−02	−	2.09E+01	7.67E−02	=	**2.03E+01**	**9.88E−02**	−	2.06E+01	5.11E−02	−	2.09E+01	4.56E−02	=
f06	2.79E−01	5.86E−01	2.74E+01	1.45E+00	+	2.14E+01	3.56E+00	+	5.16E+00	2.01E+00	+	5.19E+00	1.64E+00	+	1.39E+00	1.31E+00	+
f07	**0.00E+00**	**0.00E+00**	6.74E−07	2.39E−06	+	2.79E−02	2.74E−02	+	8.46E−04	1.59E−03	+	2.37E−02	2.31E−02	+	5.46E−03	6.61E−03	+
f08	1.59E+02	1.00E+01	2.81E+01	3.15E+00	−	9.49E+01	2.23E+01	−	**2.04E+01**	**7.04E+00**	−	5.59E+01	1.10E+01	−	8.90E+01	2.22E+01	−
f09	1.61E+02	1.04E+01	1.42E+02	1.35E+01	−	1.35E+02	3.14E+01	−	**5.80E+01**	**1.65E+01**	−	8.46E+01	9.06E+00	−	1.81E+02	1.12E+01	+
f10	6.32E+03	2.55E+02	2.74E+03	1.84E+01	−	2.27E+03	5.35E+02	−	**3.29E+02**	**2.47E+02**	−	2.17E+03	4.92E+02	−	1.71E+03	1.06E+03	−
f11	6.57E+03	3.04E+02	5.68E+03	2.43E+02	−	3.49E+03	6.80E+02	−	**2.74E+03**	**6.44E+02**	−	3.86E+03	3.52E+02	−	6.41E+03	2.93E+02	=
f12	2.29E+00	3.10E−01	1.05E+00	1.56E−01	−	1.37E+00	5.49E−01	−	**4.44E−01**	**1.66E−01**	−	9.51E−01	1.35E−01	−	2.09E+00	2.42E−01	=
f13	**2.49E−01**	**3.21E−02**	4.56E−01	4.52E−02	+	5.68E−01	1.16E−01	+	3.05E−01	5.50E−02	+	2.86E−01	6.10E−02	+	3.79E−01	4.26E−02	+
f14	2.51E−01	3.53E−02	2.65E−01	2.45E−02	+	4.44E−01	2.27E−01	+	2.36E−01	3.37E−02	=	**2.26E−01**	**4.28E−02**	−	3.90E−01	2.08E−01	+
f15	1.39E+01	7.80E−01	1.35E+01	1.24E+00	=	1.54E+01	6.70E+00	+	**5.92E+00**	**2.59E+00**	−	7.73E+00	1.04E+00	−	1.66E+01	1.26E+00	+
f16	1.19E+01	3.28E−01	1.36E+01	1.85E−01	+	1.22E+01	5.74E−01	+	1.06E+01	7.70E−01	=	**1.04E+01**	**4.58E−01**	=	1.22E+01	2.73E−01	+
f17	**4.80E+02**	**2.34E+02**	1.68E+03	1.45E+02	+	6.54E+03	8.72E+03	+	1.24E+03	3.79E+02	+	1.10E+03	4.14E+02	+	7.64E+05	2.61E+05	+
f18	7.33E+01	2.28E+01	**4.69E+01**	**6.34E+00**	−	1.23E+02	6.70E+01	+	9.54E+01	4.34E+01	+	1.10E+02	3.81E+01	+	1.68E+03	1.30E+03	+
f19	**5.37E+00**	**6.72E−01**	1.25E+01	8.54E−01	+	1.32E+01	1.18E+01	+	5.65E+00	1.46E+00	+	8.88E+00	1.21E+01	+	6.29E+00	1.28E+03	+
f20	5.77E+01	2.14E+01	**2.64E+01**	**3.45E+00**	−	1.33E+02	7.99E+01	+	3.73E+01	2.55E+01	−	3.89E+01	2.21E+01	−	1.14E+02	1.65E+01	+
f21	4.85E+02	1.96E+02	7.16E+02	1.45E+02	+	3.01E+03	3.32E+03	+	4.71E+02	2.34E+02	=	**3.86E+02**	**1.91E+02**	−	3.45E+04	1.49E+04	+
f22	2.29E+02	1.06E+02	2.49E+02	7.98E+01	+	4.57E+02	1.68E+02	+	1.91E+02	1.19E+02	−	2.31E+02	8.16E+01	+	**1.55E+02**	**1.10E+02**	−
f23	3.15E+02	6.61E−03	**3.14E+02**	**2.85E−09**	−	3.15E+02	1.24E−12	+	3.15E+02	1.40E−06	+	3.15E+02	5.74E−14	+	3.15E+02	4.02E−13	+
f24	**2.21E+02**	**7.79E+00**	2.43E+02	1.56E+01	+	2.50E+02	6.86E+00	+	2.24E+02	1.65E+00	+	2.22E+02	7.48E+00	+	2.23E+02	7.13E+00	+
f25	2.03E+02	2.74E−01	**2.00E+02**	**6.58E−04**	−	2.07E+02	4.01E+00	+	2.03E+02	1.17E−01	+	2.04E+02	3.22E+00	+	2.09E+02	1.77E+00	+
f26	**1.00E+02**	**3.24E−02**	1.00E+02	4.56E−02	+	1.01E+02	1.20E−01	+	1.00E+02	4.14E−02	+	1.39E+02	4.91E+01	+	1.00E+02	6.13E−02	+
f27	**3.34E+02**	**4.34E+01**	1.10E+03	6.71E+01	+	6.47E+02	2.50E+02	+	4.69E+02	9.46E+01	+	4.21E+02	4.64E+01	+	3.45E+02	4.44E+01	+
f28	8.47E+02	9.35E+01	**3.72E+02**	**6.37E−01**	−	1.60E+03	5.92E+02	+	9.05E+02	1.21E+02	+	9.16E+02	1.63E+02	+	7.74E+02	1.03E+02	−
f29	8.24E+02	6.91E+01	**2.16E+02**	**9.50E+01**	−	7.55E+05	2.62E+06	+	6.52E+05	2.66E+06	+	3.39E+05	2.41E+06	+	3.57E+03	1.97E+03	+
f30	1.74E+03	8.95E+02	7.36E+02	1.02E+02	−	2.98E+03	1.26E+03	+	1.70E+03	8.67E+02	=	1.29E+03	5.14E+02	−	2.24E+03	6.54E+02	+
	+/=/−		14/2/14			23/2/5			15/5/10			19/2/9			21/4/5		

Table 8
Mean and SD of best error value obtained in 51 independent runs by SS, CMA-ES, I-POP-CMAES, LS-CMAES, CMSAES, and (1+1)cholesky_CMAES on 30-*D* CEC2014 problem suite (Mean: Mean of best error, SD: Standard deviation of best error, W: result of Wilcoxon signed rank test).

Nos	SS		CMA-ES			I-POP-CMAES			LS-CMAES			CMSAES			(1+1)cholesky CMAES		
	Mean	SD	Mean	SD	W	Mean	SD	W	Mean	SD	W	Mean	SD	W	Mean	SD	W
f01	8.75E+03	6.70E+03	8.16E+04	2.66E+04	+	**0.00E+00**	**0.00E+00**	−	2.54E+08	1.37E+08	+	7.56E+08	1.40E+08	+	**0.00E+00**	**0.00E+00**	−
f02	**0.00E+00**	**0.00E+00**	4.59E+10	9.72E+09	+	**0.00E+00**	**0.00E+00**	=	6.98E+04	3.25E+04	+	6.85E+10	5.93E+09	+	**0.00E+00**	**0.00E+00**	=
f03	1.03E−07	2.81E−07	2.96E+03	1.29E+03	+	**0.00E+00**	**0.00E+00**	−	2.15E−02	1.90E−02	+	1.26E+05	1.75E+04	+	**0.00E+00**	**0.00E+00**	−
f04	**0.00E+00**	**0.00E+00**	4.71E+03	1.33E+03	+	**0.00E+00**	**0.00E+00**	=	5.50E+00	1.62E+01	+	1.01E+04	1.63E+03	+	1.27E+00	9.10E+00	+
f05	2.09E+01	5.88E−02	**2.00E+01**	**3.70E−04**	−	2.11E+01	7.69E−02	+	2.01E+01	2.18E−01	−	2.10E+01	4.38E−02	+	2.00E+01	5.85E−03	−
f06	**2.79E−01**	**5.86E−01**	6.50E+01	1.57E+00	+	9.48E+00	7.65E+00	+	2.47E+01	3.55E+00	+	3.95E+01	1.04E+00	+	5.02E+01	4.68E+00	+
f07	**0.00E+00**	**0.00E+00**	1.12E+02	1.16E+01	+	**0.00E+00**	**0.00E+00**	=	4.45E−02	1.90E−02	+	6.01E+02	5.27E+01	+	1.11E−02	9.66E−03	+
f08	1.59E+02	1.00E+01	8.98E+02	1.83E+02	+	**1.25E+02**	**1.20E+02**	−	3.41E+02	1.02E+02	+	3.67E+02	1.66E+01	+	4.38E+02	7.76E+01	+
f09	1.61E+02	1.04E+01	9.14E+02	4.24E+00	+	**3.70E+01**	**6.27E+01**	−	3.85E+02	1.84E+02	+	4.31E+02	2.35E+01	+	6.14E+02	1.37E+02	+
f10	6.32E+03	2.55E+02	**1.09E+03**	**5.78E+02**	−	3.47E+03	2.02E+03	−	3.81E+03	5.94E+02	−	7.28E+03	2.80E+02	+	4.96E+03	7.40E+02	−
f11	6.57E+03	3.04E+02	**2.16E+02**	**1.11E+02**	−	2.74E+03	2.92E+03	−	3.38E+03	5.21E+02	−	7.19E+03	2.31E+02	+	5.19E+03	8.58E+02	−
f12	2.29E+00	3.10E−01	**1.20E−01**	**1.16E+00**	−	4.39E+00	1.18E+00	+	2.16E−01	5.88E−02	−	2.29E+00	2.91E−01	+	1.47E+00	7.07E−01	−
f13	**2.49E−01**	**3.21E−02**	1.31E+00	1.70E−01	+	4.67E−01	6.16E−01	+	3.38E−01	4.88E−02	+	6.61E+00	4.79E−01	+	5.86E−01	1.44E−01	+
f14	2.51E−01	3.53E−02	1.57E+01	4.61E+00	+	8.89E−01	2.34E+00	+	**1.79E−01**	**3.12E−02**	−	2.01E+02	2.24E+01	+	4.21E−01	2.12E−01	+
f15	1.39E+01	7.80E−01	1.51E+03	1.48E+00	+	2.95E+05	2.04E+06	+	**8.21E+00**	**1.97E+00**	−	1.30E+06	4.74E+05	+	2.70E+01	1.34E+01	+
f16	1.19E+01	3.28E−01	1.62E+01	6.85E+00	+	**1.09E+01**	**1.99E+00**	−	1.28E+01	4.30E−01	+	1.31E+01	1.79E−01	+	1.43E+01	3.80E−01	+
f17	**4.80E+02**	**2.34E+02**	4.42E+03	9.19E+02	+	5.35E+06	2.59E+07	+	6.98E+04	3.19E+05	+	2.25E+07	8.63E+06	+	1.77E+03	3.91E+02	+
f18	**7.33E+01**	**2.28E+01**	2.50E+03	2.66E+02	+	7.94E+02	3.68E+03	+	4.47E+04	3.15E+05	+	9.95E+08	3.53E+08	+	1.43E+02	4.82E+01	+
f19	**5.37E+00**	**6.72E−01**	2.04E+02	2.19E+01	+	3.68E+01	1.01E+02	+	9.82E+00	1.42E+00	+	2.78E+02	3.94E+01	+	3.42E+01	4.02E+01	+
f20	**5.77E+01**	**2.14E+01**	2.76E+03	2.45E+02	+	1.92E+03	9.92E+03	+	5.40E+02	3.42E+02	+	9.63E+04	3.81E+04	+	3.84E+02	1.24E+02	+
f21	**4.85E+02**	**1.96E+02**	3.53E+03	1.22E+03	+	1.81E+06	1.24E+07	+	4.35E+04	2.65E+05	+	5.48E+06	2.46E+06	+	1.15E+03	3.80E+02	+
f22	2.29E+02	1.06E+02	5.17E+03	1.94E+02	+	**2.19E+02**	**1.00E+02**	−	4.98E+02	1.41E+02	+	1.43E+03	1.85E+02	+	8.63E+02	3.30E+02	+
f23	3.15E+02	6.61E−03	**2.81E+02**	**1.43E+01**	−	3.15E+02	2.30E−13	+	3.14E+02	8.99E+00	−	7.45E+02	7.97E+01	+	3.15E+02	4.09E−08	+
f24	2.21E+02	7.79E+00	2.76E+02	1.22E−01	+	2.28E+02	1.47E+01	+	**2.16E+02**	**2.32E+01**	−	4.06E+02	1.29E+01	+	4.78E+02	3.61E+02	+
f25	**2.03E+02**	**2.74E−01**	2.09E+02	9.14E+00	+	2.05E+02	3.61E+00	+	2.07E+02	9.31E+00	+	2.63E+02	7.69E+00	+	2.26E+02	1.61E+01	+
f26	**1.00E+02**	**3.24E−02**	2.75E+02	4.28E+01	+	1.24E+02	6.59E+01	+	1.00E+02	9.08E−02	+	1.06E+02	6.60E−01	+	1.93E+02	1.73E+02	+
f27	3.34E+02	4.34E+01	4.50E+03	8.76E+02	+	**3.02E+02**	**2.40E+00**	−	4.19E+02	1.40E+02	+	8.82E+02	8.61E+01	+	9.03E+02	4.26E+02	+
f28	**8.47E+02**	**9.35E+01**	5.98E+03	1.36E+02	+	9.72E+02	2.01E+02	+	2.54E+03	9.65E+02	+	3.58E+03	5.23E+02	+	7.88E+03	3.12E+03	+
f29	**8.24E+02**	**6.91E+01**	1.27E+07	6.28E+06	+	2.42E+05	1.71E+06	+	3.59E+03	2.89E+03	+	8.34E+07	3.53E+07	+	4.19E+06	1.17E+07	+
f30	**1.74E+03**	**8.95E+02**	7.18E+05	2.96E+05	+	4.92E+04	2.35E+05	+	7.65E+03	3.29E+03	+	8.96E+05	2.79E+05	+	4.36E+03	1.62E+03	+
	+/=/−		25/0/5			18/3/9			22/0/8			30/0/0			23/1/6		

Table 9
Mean and SD of best error value obtained in 51 independent runs by SS, GWO, GOA, MVO, and SCA on 30-*D* CEC2014 problem suite (Mean: Mean of best error, SD: Standard deviation of best error, W: result of Wilcoxon signed rank test).

Nos	SS		GWO			GOA			MVO			SCA		
	Mean	SD	Mean	SD	W	Mean	SD	W	Mean	SD	W	Mean	SD	W
1	**8.75E+03**	**6.70E+03**	4.22E+07	2.57E+07	+	1.99E+09	1.73E+09	+	1.96E+07	2.54E+07	+	1.58E+08	1.06E+08	+
2	**0.00E+00**	**0.00E+00**	1.06E+09	1.33E+09	+	8.13E+10	6.70E+10	+	4.58E+08	1.17E+09	+	1.01E+10	7.71E+09	+
3	**1.03E−07**	**2.81E−07**	2.52E+04	8.00E+03	+	1.30E+06	2.57E+06	+	9.61E+03	1.27E+04	+	3.16E+04	8.77E+03	+
4	**0.00E+00**	**0.00E+00**	1.85E+02	4.27E+01	+	2.36E+04	2.00E+04	+	1.29E+02	5.98E+01	+	6.69E+02	4.17E+02	+
5	2.09E+01	5.88E−02	2.09E+01	5.92E−02	=	2.12E+01	2.13E−01	+	2.04E+01	4.35E−01	−	2.09E+01	5.67E−02	=
6	**2.79E−01**	**5.86E−01**	1.20E+01	3.08E+00	+	3.37E+01	1.81E+01	+	1.01E+01	2.95E+00	+	2.48E+01	1.09E+01	+
7	**0.00E+00**	**0.00E+00**	7.62E+00	4.23E+00	+	7.56E+02	6.17E+02	+	3.03E+00	4.57E+00	+	8.74E+01	6.84E+01	+
8	1.59E+02	1.00E+01	**6.81E+01**	**1.99E+01**	−	3.21E+02	2.08E+02	+	7.07E+01	2.05E+01	−	1.69E+02	8.60E+01	=
9	1.61E+02	1.04E+01	**8.77E+01**	**2.62E+01**	−	4.11E+02	2.58E+02	+	9.17E+01	2.79E+01	−	2.04E+02	9.14E+01	+
10	6.32E+03	2.55E+02	1.90E+03	5.38E+02	−	6.43E+03	3.73E+03	=	2.33E+03	6.69E+02	−	4.31E+03	2.05E+03	−
11	6.57E+03	3.04E+02	**2.60E+03**	**6.05E+02**	−	6.68E+03	3.37E+03	=	2.79E+03	6.06E+02	−	5.20E+03	2.18E+03	−
12	2.29E+00	3.10E−01	1.70E+00	1.12E+00	−	4.55E+00	2.44E+00	+	**8.37E−01**	**1.06E+00**	−	2.21E+00	8.28E−01	=
13	**2.49E−01**	**3.21E−02**	3.55E−01	8.61E−02	+	6.62E+00	5.12E+00	+	4.03E−01	8.31E−02	+	1.89E+00	1.24E+00	+
14	**2.51E−01**	**3.53E−02**	5.76E−01	5.66E−01	+	2.46E+02	2.05E+02	+	5.08E−01	2.95E−01	+	2.70E+01	2.24E+01	+
15	1.39E+01	7.80E−01	6.22E+01	2.75E+02	+	1.16E+07	1.24E+07	+	**1.09E+01**	**7.81E+00**	−	1.86E+03	2.69E+03	+
16	1.19E+01	3.28E−01	1.07E+01	7.17E−01	−	1.29E+01	1.81E+00	+	1.12E+01	7.05E−01	−	1.19E+01	1.11E+00	=
17	**4.80E+02**	**2.34E+02**	1.06E+06	1.35E+06	+	1.91E+08	1.99E+08	+	4.33E+05	6.98E+05	+	4.22E+06	3.53E+06	+
18	**7.33E+01**	**2.28E+01**	5.69E+06	1.70E+07	+	5.34E+09	4.89E+09	+	1.42E+06	7.13E+06	+	8.57E+07	8.46E+07	+
19	**5.37E+00**	**6.72E−01**	2.59E+01	1.76E+01	+	6.81E+02	6.08E+02	+	1.74E+01	1.47E+01	+	6.19E+01	3.52E+01	+
20	**5.77E+01**	**2.14E+01**	1.28E+04	8.64E+03	+	9.40E+06	1.70E+07	+	5.95E+03	8.75E+03	+	1.34E+04	5.86E+03	+
21	**4.85E+02**	**1.96E+02**	5.42E+05	1.16E+06	+	9.40E+07	9.20E+07	+	2.85E+05	8.76E+05	+	1.14E+06	1.11E+06	+
22	2.29E+02	1.06E+02	3.79E+02	1.87E+02	+	1.34E+04	2.32E+04	+	3.65E+02	1.47E+02	+	6.32E+02	2.64E+02	+
23	3.15E+02	6.61E−03	3.27E+02	4.28E+00	+	1.28E+03	8.31E+02	+	3.20E+02	5.67E+00	+	3.54E+02	2.64E+01	+
24	2.21E+02	7.79E+00	**2.00E+02**	**1.01E−03**	−	4.04E+02	1.67E+02	+	2.15E+02	1.49E+01	=	2.00E+02	2.69E−01	−
25	**2.03E+02**	**2.74E−01**	2.10E+02	3.07E+00	+	3.34E+02	1.07E+02	+	2.07E+02	3.24E+00	+	2.20E+02	1.01E+01	+
26	**1.00E+02**	**3.24E−02**	1.16E+02	3.66E+01	+	2.42E+02	1.35E+02	+	1.16E+02	3.66E+01	+	1.11E+02	2.96E+01	+
27	**3.34E+02**	**4.34E+01**	5.83E+02	1.16E+02	+	1.33E+03	6.36E+02	+	5.41E+02	1.24E+02	+	6.65E+02	2.52E+02	+
28	**8.47E+02**	**9.35E+01**	9.61E+02	1.47E+02	+	5.30E+03	3.58E+03	+	9.97E+02	2.04E+02	+	1.58E+03	6.04E+02	+
29	**8.24E+02**	**6.91E+01**	8.97E+04	1.91E+05	+	4.89E+08	4.37E+08	+	9.42E+05	3.12E+06	+	8.13E+06	8.72E+06	+
30	**1.74E+03**	**8.95E+02**	2.95E+04	2.16E+04	+	6.43E+06	6.54E+06	+	1.54E+04	1.80E+04	+	1.57E+05	1.18E+05	+
	+/−/=		22/1/7			28/2/0			21/1/8			23/4/2		

Table 10
Mean and SD of best error value obtained in 51 independent runs by SS, SHO, SSA, SOA, and WOA on 30-*D* CEC2014 problem suite (Mean: Mean of best error, SD: Standard deviation of best error, W: result of Wilcoxon signed rank test).

Nos	SS		SHO			SSA			SOA			WOA		
	Mean	SD	Mean	SD	W	Mean	SD	W	Mean	SD	W	Mean	SD	W
1	**8.75E+03**	**6.70E+03**	1.80E+07	2.65E+07	+	1.94E+07	2.56E+07	+	5.58E+07	1.15E+07	+	1.97E+07	1.81E+07	+
2	**0.00E+00**	**0.00E+00**	4.58E+08	1.17E+09	+	4.58E+08	1.17E+09	+	1.21E+08	1.36E+06	+	1.26E+08	1.20E+07	+
3	**1.03E−07**	**2.81E−07**	9.99E+03	1.24E+04	+	1.15E+04	1.13E+04	+	4.04E+04	1.35E+03	+	1.87E+04	2.03E+04	+
4	**0.00E+00**	**0.00E+00**	1.19E+02	6.55E+01	+	1.34E+02	5.71E+01	+	1.58E+02	4.68E−01	+	1.19E+02	5.68E+01	+
5	2.09E+01	5.88E−02	2.04E+01	4.58E−01	−	**2.04E+01**	**4.28E−01**	−	2.09E+01	3.12E−02	=	2.09E+01	7.09E−02	−
6	**2.79E−01**	**5.86E−01**	2.20E+01	8.92E+00	+	1.63E+01	5.97E+00	+	1.24E+01	2.27E−02	+	1.74E+01	3.35E+00	+
7	**0.00E+00**	**0.00E+00**	3.00E+00	4.59E+00	+	2.99E+00	4.60E+00	+	7.65E+00	1.50E−02	+	1.88E+00	2.68E−02	+
8	1.59E+02	1.00E+01	8.80E+01	2.32E+01	−	9.90E+01	3.48E+01	−	8.16E+01	4.92E−01	−	1.48E+00	2.31E−01	−
9	1.61E+02	1.04E+01	1.25E+02	3.56E+01	−	1.13E+02	3.88E+01	−	8.68E+01	5.12E−01	−	9.63E+01	2.40E+01	−
10	6.32E+03	2.55E+02	1.98E+03	5.42E+02	−	2.99E+03	1.09E+03	−	2.08E+03	2.65E+00	−	**1.34E+03**	**5.27E+02**	−
11	6.57E+03	3.04E+02	3.37E+03	9.00E+02	−	3.34E+03	8.63E+02	−	2.11E+03	5.12E+00	−	2.46E+03	4.95E+02	−
12	2.29E+00	3.10E−01	1.05E+00	9.54E−01	−	1.02E+00	9.93E−01	−	1.54E−01	1.94E−03	−	1.56E−01	4.62E−02	−
13	**2.49E−01**	**3.21E−02**	4.38E−01	1.17E−01	+	4.51E−01	1.23E−01	+	4.44E−01	9.16E−03	+	5.33E−01	1.27E−01	+
14	**2.51E−01**	**3.53E−02**	3.88E−01	2.16E−01	+	4.49E−01	2.52E−01	+	7.34E−01	4.21E−02	+	3.37E−01	1.03E−01	+
15	1.39E+01	7.80E−01	3.47E+01	1.92E+01	+	1.12E+01	7.76E+00	−	1.91E+01	1.02E−01	+	1.75E+01	1.02E+01	+
16	1.19E+01	3.28E−01	1.18E+01	1.02E+00	=	1.14E+01	8.39E−01	−	**1.06E+01**	**1.57E−03**	−	1.12E+01	6.77E−01	−
17	**4.80E+02**	**2.34E+02**	4.17E+05	7.01E+05	+	4.76E+05	6.84E+05	+	6.52E+05	8.23E+04	+	1.34E+06	2.13E+06	+
18	**7.33E+01**	**2.28E+01**	1.41E+06	7.13E+06	+	1.42E+06	7.13E+06	+	1.23E+04	3.84E+01	+	1.99E+03	3.70E+03	+
19	**5.37E+00**	**6.72E−01**	2.18E+01	1.69E+01	+	1.92E+01	1.38E+01	+	1.23E+01	5.87E−03	+	2.67E+01	3.44E+01	+
20	**5.77E+01**	**2.14E+01**	8.22E+03	7.59E+03	+	6.09E+03	8.66E+03	+	8.75E+03	5.84E+01	+	5.84E+03	1.93E+00	+
21	**4.85E+02**	**1.96E+02**	2.81E+05	8.78E+05	+	3.00E+05	8.74E+05	+	6.95E+05	2.15E+04	+	8.60E+05	7.85E+05	+
22	2.29E+02	1.06E+02	5.73E+02	2.48E+02	+	3.91E+02	1.62E+02	+	3.65E+02	2.56E+01	+	5.69E+02	2.06E+02	+
23	3.15E+02	6.61E−03	3.19E+02	5.88E+00	+	3.20E+02	5.67E+00	+	3.54E+02	1.15E−01	+	3.16E+02	1.01E+00	+
24	2.21E+02	7.79E+00	2.17E+02	1.38E+01	=	2.17E+02	1.55E+01	=	**2.00E+02**	**2.34E−04**	−	2.15E+02	5.15E+00	+
25	**2.03E+02**	**2.74E−01**	2.11E+02	3.43E+00	+	2.11E+02	3.70E+00	+	2.09E+02	5.96E−03	+	2.11E+02	4.59E+00	+
26	**1.00E+02**	**3.24E−02**	1.49E+02	4.93E+01	+	1.10E+02	2.99E+01	+	1.00E+02	2.02E−02	=	1.51E+02	5.13E+01	+
27	**3.34E+02**	**4.34E+01**	7.31E+02	3.32E+02	+	6.40E+02	1.53E+02	+	5.62E+02	1.04E+00	+	7.62E+02	1.98E+02	+
28	**8.47E+02**	**9.35E+01**	2.53E+03	1.47E+03	+	9.85E+02	1.41E+02	+	1.72E+03	4.63E−01	+	1.51E+03	4.53E+02	+
29	**8.24E+02**	**6.91E+01**	5.78E+05	2.14E+06	+	1.05E+06	3.49E+06	+	9.68E+03	1.43E+01	+	8.68E+05	2.67E+06	+
30	**1.74E+03**	**8.95E+02**	1.34E+04	1.90E+04	+	1.77E+04	1.73E+04	+	3.16E+04	6.56E+02	+	1.38E+04	1.98E+02	+
	+/−/=		22/2/6			21/1/8			21/2/7			23/0/7		

Table 11
Ranking of algorithm according to Friedman ranking based on mean error value. (FR: Friedman Ranking).

S.N.	Algorithm	FR	Rank	S.N.	Algorithm	FR	Rank
1	**SS**	**5.4667**	**1**	13	I-POP-CMAES	11.4000	10
2	PSO	15.5667	19	14	LS-CMAES	10.6500	9
3	BB-PSO	8.4833	6	15	CMSAES	21.4333	23
4	CLPSO	7.4167	4	16	(1+1)cholesky CMAES	14.0167	18
5	APSO	18.2333	21	17	GWO	14.0000	16
6	OLPSO	7.1000	3	18	GOA	22.5500	24
7	CoBiDE	8.7833	7	19	MVO	12.4500	12
8	FCDE	11.6667	11	20	SCA	18.3167	22
9	RSDE	5.6500	2	21	SHO	14.0000	17
10	POBL_ADE	7.4833	5	22	SSA	13.8333	15
11	DE_best	9.7000	8	23	SOA	12.6500	13
12	CMA-ES	15.8333	20	24	WOA	13.3167	14

exploitative and exploratory search, the half population of better solution utilizes *towards-rand* to calculate search direction and rest half of the population uses *towards-best* to calculate the search direction thereby driving diversity in the set of better solutions and pushing the inferior solutions to attempt for improved objective values. In addition to this approach, three other combinations of *towards-rand*, and *towards-best* can be incorporated in SS. To analyze the effectiveness of the proposed approach, the following combinations have been compared with the SS on 30D problems of IEEE CEC 2014 problem suite.

1. SStr: SS with only *towards-rand*,
2. SStb: SS with only *towards-best*, and
3. SSr: SS with random selection of *towards-rand* and *towards-best* with equal probability.

In Table 2, the outcomes of the aforementioned configuration of SS with original SS on 30D problems of IEEE CEC 2014 suite are reported. The aim of this experiment is to test the effectiveness of SS in terms of a balance between exploitative and exploratory search. The Wilcoxon's signed rank test (W) is utilized to check the significance of the results of SS with a comparison of other configurations of SS. In Table 2, the row summarizes the outcomes of WT in terms of the number of superior, equal and inferior performance of SS compared to other respective algorithms. The results show that the SS outperforms the SStr, SStb, and SSr on 11, 22, and 14 out of 30 problems, respectively. Furthermore, SS is outperformed by SStr, SStb, and SSr on two, two, and five out of 30 problems, respectively. From this analysis, it can be concluded that the proposed approach provides a better balance between exploitation and exploration of search space during the optimization process. Some important outcomes of this experiment are summarized as follows.

1. Configuration SStr is the least effective as compared with other configuration because it emphasizes more on exploratory search which slows down the optimization process.
2. Configuration SStb provides a better balance between exploration and exploitation than SStr because it explores solutions in the regions of search space containing better solutions in place of whole search space.
3. To improve the balancing characteristics of SS, SSr can be a good choice because it improves the performance of the algorithm on composite problems but meanwhile it losses its performance on simpler hybrid problems. Thus, the performance of SSr is almost similar to the SStb which shows that this configuration cannot utilize the features of *towards-rand* effectively during the run.
4. Proposed configuration of SS provides performance better than the other configurations which shows that this configuration utilizes *towards-rand* and *towards-best* effectively during the optimization process and provide better balance between exploitation and exploration.

4.3. Experiment 3: SS vs state-of-art algorithms

In this experiment, to analyze the performance, SS is benchmarked on 30 real-parameter single objective bound-constrained optimization problems used in a special session of IEEE CEC 2014 [32]. Detailed information and characteristics of these problems are available in [32]. To evaluate the performance of SS on the CEC 2014 problem suite, the results are compared with other state-of-the-art algorithms. State-of-the-art algorithms are divided into four groups:

1. *Group-I:-* Variants of PSO: Basic PSO [8], BB-PSO [33], CLPSO [34], APSO [35], OLPSO [36].
2. *Group-II:-* Variants of DE: CoBiDE [37], FCDE [38], RSDE [39], POBLADE [40], DE-best [7].
3. *Group-III:-* Variants of CMA-ES: Basic CMA-ES [41], I-POP-CMAES [42], LS-CMAES [43], CMSAES [44], (1+1)cholesky-CMAES [45].
4. *Group-IV:-* Recently proposed optimization algorithms: GWO [11], GOA [21], MVO [46], SCA [47], SHO [48], SSA [22], SOA [13], WOA [14].

PSO, DE, and CMA-ES are the popular classical metaheuristics. Popular variants of classical algorithms are also taken from the literature to show the effectiveness of the SS.

In this experiment, the population size, N, is set to 80, the dimension of search space for all problems, D, is fixed to 30, and allowed maximum function evaluation, MaxFES, is fixed to 300,000 for 51 independent runs. The parameters of other algorithms are set to their default values as reported in their referred paper.

Tables 6–10 summarize the mean and standard deviation (SD) of the error values obtained by the algorithms over 51 independent runs for each problem. We also performed the Wilcoxon signed ranks test (W) in this experiment. The statistical results are also summarized in Tables 6–10, where '+' denotes the performance of the SS is better than other algorithms, '−' denotes the performance of other method is better than the SS, and '=' denotes that there is no significant difference in performance. We also rank all algorithms along with the SS using Friedman Ranking (FR) test based on the mean of error values obtained by the algorithms over 51 independent runs. The statistical results of the Friedman test are reported in Table 11.

1. *SS vs Group-I's algorithms:* Table 6 summarizes the results obtained by SS algorithm, and Group-I's algorithms: PSO, BB-PSO, CLPSO, APSO, and OLPSO on the CEC 2014 problem suite. It is seen from Table 6 that the SS is showing better performance than PSO, BB-PSO, CLPSO, APSO and OLPSO on 23, 21, 20, 23, and 18 problems out of 30, respectively. Performance of SS is worse than PSO, BB-PSO, CLPSO, APSO and OLPSO on five, eight, eight, seven, and seven problems

Table 12
Mean and SD of best error value obtained in 51 independent runs by SASS, iCMAES-ILS, NBIPOP-aCMA-ES, SHADE, LSHADE, EBOwithCMAR, and HSES on 10-*D* CEC2014 problem suite (Mean: Mean of best error, SD: Standard deviation of best error, W: result of Wilcoxon signed rank test).

Prob	SASS		iCMAES-ILS			NBIPOP-aCMA-ES			SHADE		
	Mean	SD	Mean	SD	W	Mean	SD	W	Mean	SD	W
F1	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=
F2	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=
F3	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=
F4	3.00E+01	1.21E+01	1.44E+01	1.60E+01	−	**2.82E+00**	8.24E+00	−	2.74E+01	1.43E+01	=
F5	1.45E+01	8.53E+00	1.47E+01	8.88E+00	=	1.81E+01	6.02E+00	+	1.80E+01	5.12E+00	+
F6	**0.00E+00**	**0.00E+00**	**0.00E+00**	0.00E+00	=	3.30E−01	5.83E−01	+	**0.00E+00**	**0.00E+00**	=
F7	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	0.00E+00	=	9.78E−03	1.47E−02	+
F8	**0.00E+00**	0.00E+00	2.54E−01	4.81E−01	+	3.70E+00	1.99E+00	+	**0.00E+00**	**0.00E+00**	=
F9	2.49E+00	8.03E−01	9.75E−02	2.99E−01	−	3.26E−01	6.41E−01	−	3.14E+00	8.99E−01	+
F10	2.45E−03	1.22E−02	1.22E+02	8.73E+01	+	9.16E+01	7.51E+01	+	3.67E−03	1.48E−02	=
F11	3.50E+01	4.37E+01	**8.59E+00**	8.36E+00	−	1.17E+02	1.25E+02	+	6.32E+01	5.37E+01	+
F12	6.59E−02	1.66E−02	6.50E−02	1.18E−01	=	**1.01E−02**	1.65E−02	−	1.42E−01	2.34E−02	+
F13	5.52E−02	1.11E−02	**9.11E−03**	4.40E−03	−	1.09E−02	6.21E−03	−	7.40E−02	1.43E−02	+
F14	**7.72E−02**	2.58E−02	1.55E−01	4.45E−02	+	2.82E−01	8.50E−02	+	1.06E−01	3.15E−02	+
F15	3.74E−01	6.64E−02	7.23E−01	1.93E−01	+	5.47E−01	1.28E−01	+	5.05E−01	8.84E−02	+
F16	1.31E+00	2.38E−01	1.91E+00	5.39E−01	+	2.53E+00	5.58E−01	+	1.56E+00	3.19E−01	+
F17	**7.41E−01**	7.81E−01	2.10E+01	1.71E+01	+	3.89E+01	3.97E+01	+	1.28E+00	1.92E+00	=
F18	1.41E−01	1.62E−01	5.26E−01	4.92E−01	+	3.58E+00	6.18E+00	+	2.26E−01	1.81E−01	=
F19	1.03E−01	5.37E−02	7.08E−01	5.94E−01	+	8.28E−01	5.02E−01	+	2.26E−01	2.17E−01	+
F20	**9.92E−02**	5.85E−02	8.04E−01	5.01E−01	+	1.32E+00	2.28E+00	+	2.77E−01	1.49E−01	+
F21	**3.36E−01**	2.66E−01	3.21E+00	5.51E+00	+	1.67E+01	3.02E+01	+	3.76E−01	2.94E−01	=
F22	9.63E−02	3.48E−02	1.83E+01	7.04E+00	+	1.79E+01	6.76E+00	+	2.90E−01	1.13E−01	+
F23	3.29E+02	1.72E−13	**2.50E+02**	1.41E+02	−	3.16E+02	5.58E+01	−	3.29E+02	2.87E−13	=
F24	1.08E+02	1.48E+00	1.04E+02	4.48E+00	−	1.07E+02	2.98E+00	=	1.09E+02	1.37E+00	+
F25	1.28E+02	3.40E+01	1.26E+02	1.02E+01	=	1.39E+02	2.58E+01	+	1.50E+02	4.38E+01	+
F26	**1.00E+02**	1.81E−02	1.00E+02	4.42E−03	=	1.00E+02	5.56E−03	=	1.00E+02	1.53E−02	+
F27	2.31E+01	8.75E+01	**1.05E+02**	1.37E+02	+	2.02E+02	1.36E+02	+	1.50E+02	1.68E+02	+
F28	3.68E+02	4.88E+00	3.61E+02	5.18E+01	+	3.58E+02	5.34E+01	=	3.99E+02	4.75E+01	+
F29	2.22E+02	5.05E−01	2.22E+02	5.88E−01	+	2.20E+02	7.11E+00	=	2.22E+02	5.91E−01	=
F30	**4.64E+02**	7.45E+00	4.71E+02	1.77E+01	+	4.83E+02	2.30E+01	+	4.73E+02	2.23E+01	+
	+/−/=		15/9/6			17/8/5			17/13/0		

Prob	SASS		LSHADE			EBOwithCMAR			HSES		
	Mean	SD	Mean	SD	W	Mean	SD	W	Mean	SD	W
F1	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	0.00E+00	=
F2	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	0.00E+00	=
F3	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	0.00E+00	=
F4	3.00E+01	1.21E+01	3.08E+01	1.11E+01	+	8.10E+00	1.42E+01	−	3.48E+01	5.20E−03	+
F5	1.45E+01	8.53E+00	1.77E+01	6.24E+00	+	**1.29E+01**	9.31E+00	=	1.73E+01	6.95E+00	+
F6	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	0.00E+00	=	3.09E−02	2.21E−01	=
F7	**0.00E+00**	**0.00E+00**	5.31E−03	1.13E−02	+	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	0.00E+00	=
F8	**0.00E+00**	0.00E+00	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	0.00E+00	=	3.51E−01	5.91E−01	+
F9	2.49E+00	8.03E−01	3.08E+00	1.13E+00	+	**0.00E+00**	0.00E+00	−	8.19E−01	8.37E−01	−
F10	2.45E−03	1.22E−02	4.90E−02	6.16E−02	+	**1.22E−03**	8.75E−03	=	6.11E+01	8.79E+01	+
F11	3.50E+01	4.37E+01	5.49E+01	5.83E+01	+	3.44E+01	3.98E+01	=	3.26E+01	6.45E+01	−
F12	6.59E−02	1.66E−02	5.29E−02	2.24E−02	−	2.74E−02	2.03E−02	−	3.41E−02	6.20E−02	−
F13	5.52E−02	1.11E−02	4.89E−02	1.36E−02	−	2.67E−02	1.29E−02	−	9.78E−03	3.78E−03	−
F14	**7.72E−02**	2.58E−02	9.01E−02	2.66E−02	+	1.27E−01	4.91E−02	+	3.38E−01	7.95E−02	+
F15	3.74E−01	6.64E−02	4.03E−01	8.92E−02	+	**3.15E−01**	6.63E−02	=	9.10E−01	2.35E−01	+
F16	1.31E+00	2.38E−01	1.34E+00	3.09E−01	=	**9.36E−01**	3.54E−01	=	1.59E+00	4.95E−01	+
F17	**7.41E−01**	7.81E−01	3.38E+00	4.65E+00	+	2.06E+01	3.89E+01	+	6.03E+01	1.46E+02	+
F18	1.41E−01	1.62E−01	4.75E−01	5.72E−01	+	**1.23E−01**	1.47E−01	=	3.51E−01	2.67E−01	+
F19	1.03E−01	5.37E−02	2.05E−01	2.79E−01	+	**9.36E−02**	2.32E−01	=	7.03E−01	5.38E−01	+
F20	**9.92E−02**	5.85E−02	2.73E−01	2.20E−01	+	1.75E−01	1.52E−01	+	1.61E+00	3.59E+00	+
F21	**3.36E−01**	2.66E−01	5.28E−01	2.85E−01	+	9.41E−01	3.27E+00	+	1.79E+01	6.13E+01	+
F22	9.63E−02	3.48E−02	6.97E−02	6.65E−02	−	**6.43E−02**	5.16E−02	−	2.71E+01	3.69E+01	+
F23	3.29E+02	1.72E−13	3.29E+02	2.87E−13	=	3.04E+02	5.19E+01	−	3.29E+02	2.87E−13	=
F24	1.08E+02	1.48E+00	1.09E+02	1.76E+00	+	**1.03E+02**	3.37E+00	−	1.06E+02	4.47E+00	=
F25	1.28E+02	3.40E+01	1.28E+02	3.19E+01	=	**1.17E+02**	5.03E+00	=	1.97E+02	8.70E+00	+
F26	**1.00E+02**	1.81E−02	1.00E+02	1.65E−02	=	1.00E+02	1.77E−02	=	1.00E+02	4.29E−03	=
F27	2.31E+01	8.75E+01	4.26E+01	1.15E+02	+	5.33E+01	1.23E+02	+	2.66E+02	1.08E+02	+
F28	3.68E+02	4.88E+00	3.81E+02	3.32E+01	+	3.83E+02	3.32E+01	+	**3.53E+02**	1.39E+02	=
F29	2.22E+02	5.05E−01	2.22E+02	5.62E−01	=	2.23E+02	1.14E+00	+	**2.19E+02**	1.89E+01	−
F30	**4.64E+02**	7.45E+00	4.65E+02	9.02E+00	+	4.69E+02	1.45E+01	+	5.48E+02	5.84E+01	+
	+/−/=		17/10/3			8/15/7			16/9/5		

respectively, and SS is significantly equal to the PSO, BB-PSO, CLPSO, and OLPSO for 2, 1, 2, and 5 problems of CEC 2014 problem suite respectively.

2. *SS vs Group-II's algorithms:* Table 7 summarizes the results of the SS algorithm, and Group-II's algorithms: CoBiDE, FCDE, RSDE, POBL_ADE, and DE_best. As shown in Table 7, SS has better performance than CoBiDE, FCDE, RSDE, POBL_ADE, and DE_best on 14, 23, 15, 19, and 21 problems out of 30 problems respectively. Performance of SS is worse than CoBiDE, FCDE, RSDE, POBL_ADE, and DE_best

Table 13
Mean and SD of best error value obtained in 51 independent runs by SASS, iCMAES-ILS, NBIPOP-aCMA-ES, SHADE, LSHADE, EBOwithCMAR, and HSES on 30-*D* CEC2014 problem suite (Mean: Mean of best error, SD: Standard deviation of best error, W: result of Wilcoxon signed rank test).

Prob	SASS		iCMAES-ILS			NBIPOP-aCMA-ES			SHADE		
	Mean	SD	Mean	SD	W	Mean	SD	W	Mean	SD	W
F1	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=	4.81E+02	7.86E+02	+
F2	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=
F3	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=
F4	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=
F5	2.01E+01	2.44E−02	**2.00E+01**	**6.78E−06**	−	2.05E+01	4.43E−01	+	2.01E+01	1.91E−02	−
F6	3.26E−04	2.30E−03	4.00E−03	2.85E−02	+	7.14E−01	1.28E+00	+	5.29E−01	7.22E−01	+
F7	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=	4.83E−04	1.97E−03	=
F8	**0.00E+00**	0.00E+00	2.42E+00	1.39E+00	+	9.98E+00	3.54E+00	+	**0.00E+00**	**0.00E+00**	=
F9	6.99E+00	1.66E+00	2.57E+00	1.24E+00	−	3.24E+00	1.47E+00	−	1.58E+01	3.17E+00	+
F10	**1.22E−03**	**4.95E−03**	1.45E+02	2.87E+02	+	6.36E+02	3.83E+02	+	1.27E−02	1.67E−02	+
F11	1.20E+03	2.40E+02	**7.38E+01**	**9.85E+01**	−	7.31E+02	4.48E+02	−	1.49E+03	1.93E+02	+
F12	1.54E−01	2.63E−02	2.83E−02	4.58E−02	−	**1.32E−02**	**6.78E−03**	−	1.65E−01	1.85E−02	=
F13	1.16E−01	1.57E−02	**2.95E−02**	**7.22E−03**	−	3.89E−02	1.26E−02	−	2.06E−01	3.64E−02	+
F14	2.38E−01	3.08E−02	**1.70E−01**	**1.99E−02**	−	3.28E−01	5.24E−02	+	2.30E−01	3.03E−02	=
F15	2.13E+00	2.95E−01	2.51E+00	4.06E−01	+	2.14E+00	3.87E−01	+	2.57E+00	3.14E−01	+
F16	8.62E+00	4.28E−01	1.09E+01	9.95E−01	+	1.06E+01	1.18E+00	+	9.19E+00	3.35E−01	+
F17	2.60E+02	1.52E+02	1.05E+03	2.91E+02	+	8.52E+02	2.96E+02	+	1.06E+03	3.18E+02	+
F18	8.19E+00	3.48E+00	9.61E+01	3.23E+01	+	1.15E+02	2.67E+01	+	6.15E+01	3.40E+01	+
F19	3.65E+00	5.79E−01	6.46E+00	1.25E+00	+	5.70E+00	1.58E+00	+	4.43E+00	7.12E−01	+
F20	3.06E+00	1.39E+00	3.35E+01	4.49E+01	+	2.40E+01	3.57E+01	+	1.28E+01	7.69E+00	+
F21	9.67E+01	7.93E+01	6.44E+02	2.24E+02	+	4.91E+02	1.81E+02	+	2.63E+02	1.30E+02	+
F22	2.82E+01	1.73E+01	1.30E+02	8.16E+01	+	1.41E+02	5.16E+01	+	1.14E+02	5.57E+01	+
F23	**3.15E+02**	**2.88E−13**	**3.15E+02**	**1.72E−13**	=	**3.15E+02**	**1.72E−13**	=	**3.15E+02**	**1.72E−13**	=
F24	2.26E+02	3.43E+00	2.15E+02	1.38E+01	−	**2.12E+02**	**1.31E+01**	−	2.26E+02	3.34E+00	=
F25	**2.03E+02**	**1.35E−01**	2.03E+02	3.70E−02	=	2.03E+02	2.43E−02	=	2.03E+02	7.37E−01	+
F26	**1.00E+02**	**2.08E−02**	1.00E+02	7.85E−03	=	1.00E+02	6.69E−02	=	1.00E+02	3.54E−02	+
F27	**3.00E+02**	**0.00E+00**	**3.00E+02**	**0.00E+00**	=	**3.00E+02**	**0.00E+00**	=	3.25E+02	3.89E+01	+
F28	8.24E+02	2.18E+01	8.75E+02	2.17E+01	+	**7.96E+02**	**4.83E+01**	−	8.40E+02	3.16E+01	+
F29	7.17E+02	4.06E+00	7.27E+02	2.51E+02	+	6.66E+02	1.03E+02	=	7.28E+02	1.25E+01	+
F30	**8.94E+02**	**3.60E+02**	2.19E+03	6.27E+02	+	1.66E+03	3.83E+02	+	1.77E+03	7.68E+02	+
	+/−/=		14/9/7			14/9/7			20/9/1		

Prob	SASS		LSHADE			EBOwithCMAR			HSES		
	Mean	SD	Mean	SD	W	Mean	SD	W	Mean	SD	W
F1	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	=	3.72E−05	4.20E−05	+	**0.00E+00**	**0.00E+00**	=
F2	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=
F3	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=
F4	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=
F5	2.01E+01	2.44E−02	2.01E+01	3.68E−02	+	**2.00E+01**	**4.57E−04**	−	**2.00E+01**	**3.44E−04**	−
F6	3.26E−04	2.30E−03	**1.38E−07**	**9.89E−07**	=	1.64E−01	1.17E+00	+	1.08E+00	1.01E+00	+
F7	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=
F8	**0.00E+00**	0.00E+00	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=	7.76E+00	2.23E+00	+
F9	6.99E+00	1.66E+00	6.78E+00	1.48E+00	=	**2.06E+00**	**1.34E+00**	−	6.96E+00	2.30E+00	=
F10	**1.22E−03**	**4.95E−03**	1.63E−02	1.58E−02	+	2.00E−02	2.46E−02	+	3.49E+02	1.96E+02	+
F11	1.20E+03	2.40E+02	1.23E+03	1.83E+02	+	1.20E+03	2.27E+02	+	7.43E+02	3.24E+02	−
F12	1.54E−01	2.63E−02	1.61E−01	2.29E−02	+	6.21E−02	3.55E−02	−	1.80E−02	2.24E−02	−
F13	1.16E−01	1.57E−02	1.24E−01	1.75E−02	+	1.07E−01	2.71E−02	−	4.10E−02	9.68E−03	−
F14	2.38E−01	3.08E−02	2.42E−01	2.98E−02	+	1.97E−01	2.10E−02	−	3.43E−01	6.60E−02	+
F15	2.13E+00	2.95E−01	2.15E+00	2.51E−01	+	**1.97E+00**	**3.99E−01**	=	2.83E+00	7.23E−01	+
F16	8.62E+00	4.28E−01	**8.50E+00**	**4.58E−01**	=	8.62E+00	4.92E−01	=	9.94E+00	7.85E−01	+
F17	2.60E+02	1.52E+02	1.88E+02	7.50E+01	−	1.90E+02	9.50E+01	−	**5.34E+01**	**9.20E+01**	−
F18	8.19E+00	3.48E+00	**5.91E+00**	**2.89E+00**	−	7.02E+00	2.84E+00	=	6.84E+00	4.24E+00	−
F19	3.65E+00	5.79E−01	3.68E+00	6.80E−01	+	**2.64E+00**	**7.96E−01**	−	2.94E+00	8.01E−01	=
F20	3.06E+00	1.39E+00	3.08E+00	1.47E+00	+	3.14E+00	1.21E+00	+	**2.31E+00**	**1.99E+00**	=
F21	9.67E+01	7.93E+01	8.68E+01	8.99E+01	=	1.26E+02	9.54E+01	+	**3.14E+01**	**7.48E+01**	−
F22	2.82E+01	1.73E+01	**2.76E+01**	**1.79E+01**	=	7.36E+01	5.80E+01	+	1.76E+02	7.65E+01	+
F23	**3.15E+02**	**2.88E−13**	**3.15E+02**	**1.72E−13**	=	**3.15E+02**	**5.74E−14**	=	3.15E+02	6.16E−06	+
F24	2.26E+02	3.43E+00	2.24E+02	1.06E+00	−	2.22E+02	4.48E+00	=	2.24E+02	9.38E−01	=
F25	**2.03E+02**	**1.35E−01**	2.03E+02	4.96E−02	=	2.03E+02	8.06E−02	+	2.08E+02	2.20E+00	+
F26	**1.00E+02**	**2.08E−02**	1.00E+02	1.55E−02	=	1.00E+02	3.23E−02	=	1.40E+02	4.25E+01	+
F27	**3.00E+02**	**0.00E+00**	**3.00E+02**	**0.00E+00**	=	3.55E+02	5.03E+01	+	3.01E+02	5.93E+00	+
F28	8.24E+02	2.18E+01	8.40E+02	1.40E+01	+	8.43E+02	1.67E+01	+	8.96E+02	2.39E+01	+
F29	7.17E+02	4.06E+00	7.17E+02	5.13E+00	=	7.17E+02	3.92E+00	=	**2.93E+02**	**7.96E+01**	−
F30	**8.94E+02**	**3.60E+02**	1.25E+03	6.20E+02	+	9.41E+02	3.32E+02	+	1.70E+03	3.70E+02	+
	+/−/=		11/16/3			11/12/7			13/9/8		

on 14, five, 10, nine, and five problems respectively, and SS provides performance similar to the CoBiDE, FCDE, RSDE, POBL_ADE, and DE_best for two, two, five, two and seven problems of CEC 2014 problem suite respectively.

3. *SS vs Group-III's algorithms:* Table 8 presents the results of the SS algorithm, and Group-III's algorithms: CMA-ES, I-POP-CMAES, LS-CMAES, CMSAES, and (1+1)cholesky-CMAES. Table 8 shows that SS performs better than CMA-ES, I-POP-CMAES, LS-CMAES, CMSAES, and (1+1)Cholesky-CMAES

Table 14
Mean and SD of best error value obtained in 51 independent runs by SASS, iCMAES-ILS, NBIPOP-aCMA-ES, SHADE, LSHADE, EBOwithCMAR, and HSES on 50-*D* CEC2014 problem suite (Mean: Mean of best error, SD: Standard deviation of best error, W: result of Wilcoxon signed rank test).

Prob	SASS		iCMAES-ILS			NBIPOP-aCMA-ES			SHADE		
	Mean	SD	Mean	SD	W	Mean	SD	W	Mean	SD	W
F1	3.16E+02	6.46E+02	**0.00E+00**	**0.00E+00**	−	**0.00E+00**	**0.00E+00**	−	1.86E+04	1.36E+04	+
F2	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=
F3	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=
F4	1.37E+01	2.96E+01	1.03E+01	2.44E+01	=	0.00E+00	0.00E+00	−	9.72E+00	2.94E+01	−
F5	2.03E+01	3.05E−02	2.00E+01	6.43E−05	−	2.08E+01	5.20E−01	+	2.01E+01	2.02E−02	−
F6	6.61E−01	6.61E−01	2.24E−04	9.56E−04	−	1.31E+00	2.21E+00	+	5.17E+00	2.43E+00	+
F7	1.45E−04	1.04E−03	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=	3.91E−03	7.41E−03	+
F8	**0.00E+00**	**0.00E+00**	5.76E+00	2.01E+00	+	2.52E+01	8.83E+00	+	**0.00E+00**	**0.00E+00**	=
F9	1.31E+01	1.85E+00	6.37E+00	1.87E+00	−	5.82E+00	2.96E+00	−	3.42E+01	6.16E+00	+
F10	3.80E−02	2.08E−02	1.60E+02	2.52E+02	+	2.04E+03	7.74E+02	+	**1.03E−02**	**1.14E−02**	−
F11	3.36E+03	3.01E+02	**1.13E+02**	**1.27E+02**	−	1.67E+03	8.77E+02	−	3.56E+03	3.27E+02	+
F12	2.20E−01	2.69E−02	1.03E−02	3.03E−02	−	1.31E−02	6.85E−03	−	1.64E−01	2.12E−02	−
F13	1.71E−01	1.73E−02	6.60E−02	1.86E−02	−	7.97E−02	2.95E−02	−	3.10E−01	4.55E−02	+
F14	3.07E−01	2.11E−02	2.43E−01	3.25E−02	=	3.18E−01	5.56E−02	+	2.95E−01	5.72E−02	−
F15	5.23E+00	4.30E−01	4.56E+00	4.07E−01	=	**4.44E+00**	**5.55E−01**	−	5.74E+00	6.15E−01	+
F16	1.70E+01	4.47E−01	1.99E+01	7.20E−01	+	1.89E+01	1.79E+00	+	1.74E+01	4.27E−01	+
F17	1.75E+03	3.48E+02	2.12E+03	4.02E+02	+	1.85E+03	2.83E+02	+	2.23E+03	7.81E+02	+
F18	1.01E+02	1.16E+01	1.72E+02	8.30E+01	+	2.47E+02	4.84E+01	+	1.54E+02	3.66E+01	+
F19	9.30E+00	1.74E+00	1.44E+01	2.29E+00	+	1.24E+01	2.12E+00	+	9.38E+00	3.25E+00	=
F20	1.43E+01	4.85E+00	2.93E+02	8.50E+01	+	2.43E+02	6.79E+01	+	1.98E+02	5.73E+01	+
F21	6.34E+02	2.40E+02	1.54E+03	3.31E+02	+	1.45E+03	3.06E+02	+	1.23E+03	4.06E+02	+
F22	1.32E+02	7.59E+01	2.01E+02	1.63E+02	+	2.71E+02	1.39E+02	+	3.68E+02	1.59E+02	+
F23	**3.44E+02**	**3.59E−13**	**3.44E+02**	**5.74E−14**	=	**3.44E+02**	**5.74E−14**	=	**3.44E+02**	**5.74E−14**	=
F24	2.76E+02	5.73E−01	**2.62E+02**	**2.86E+00**	−	2.71E+02	2.63E+00	−	2.75E+02	1.73E+00	−
F25	**2.05E+02**	**4.38E−01**	2.05E+02	3.05E−01	=	2.05E+02	2.70E−01	=	2.12E+02	6.92E+00	+
F26	**1.00E+02**	**1.94E−02**	**1.00E+02**	**1.32E−01**	=	**1.00E+02**	**1.01E−01**	=	1.04E+02	1.95E+01	+
F27	3.29E+02	3.07E+01	3.10E+02	1.73E+01	−	3.17E+02	2.23E+01	=	4.53E+02	5.70E+01	+
F28	**1.12E+03**	**2.78E+01**	1.25E+03	7.98E+01	+	1.15E+03	7.54E+01	+	1.14E+03	5.26E+01	+
F29	7.98E+02	3.42E+01	8.05E+02	4.74E+01	+	7.94E+02	2.96E+01	=	8.91E+02	5.54E+01	+
F30	8.80E+03	4.38E+02	9.48E+03	6.10E+02	+	1.04E+04	6.38E+02	+	9.38E+03	7.87E+02	+
	+/−/=		13/9/8			14/8/8			19/5/6		

Prob	SASS		LSHADE			EBOwithCMAR			HSES		
	Mean	SD	Mean	SD	W	Mean	SD	W	Mean	SD	W
F1	3.16E+02	6.46E+02	1.24E+03	1.52E+03	+	6.25E−03	1.98E−03	−	3.61E−09	1.74E−08	−
F2	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=	3.82E−10	2.73E−09	+
F3	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=	1.23E−09	5.35E−09	+
F4	1.37E+01	2.96E+01	5.89E+01	4.56E+01	+	2.89E+01	4.51E+01	+	**0.00E+00**	**0.00E+00**	−
F5	2.03E+01	3.05E−02	2.03E+01	3.09E−02	=	**2.00E+01**	**1.46E−04**	−	**2.00E+01**	**1.12E−04**	−
F6	6.61E−01	6.61E−01	3.58E−01	5.44E−01	−	1.06E−02	7.28E−02	=	**3.23E−05**	7.61E−05	−
F7	1.45E−04	1.04E−03	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=	6.19E−10	3.29E−09	=
F8	**0.00E+00**	**0.00E+00**	2.58E−09	7.48E−09	+	**0.00E+00**	**0.00E+00**	=	3.57E+00	2.29E+00	+
F9	1.31E+01	1.85E+00	1.15E+01	2.06E+00	−	8.03E+00	2.86E+00	−	**2.19E+00**	**1.57E+00**	−
F10	3.80E−02	2.08E−02	1.22E−01	4.13E−02	+	2.96E−01	2.57E−01	+	5.27E+02	3.76E+02	+
F11	3.36E+03	3.01E+02	3.26E+03	3.17E+02	=	3.23E+03	4.48E+02	=	9.18E+02	3.79E+02	−
F12	2.20E−01	2.69E−02	2.23E−01	2.34E−02	+	5.39E−02	4.39E−02	−	**1.20E−03**	**1.70E−03**	−
F13	1.71E−01	1.73E−02	1.62E−01	1.76E−02	−	1.81E−01	4.54E−02	+	**4.36E−02**	**8.00E−03**	−
F14	3.07E−01	2.11E−02	3.11E−01	2.23E−02	+	**2.24E−01**	**2.29E−02**	−	3.85E−01	4.45E−02	+
F15	5.23E+00	4.30E−01	5.15E+00	5.08E−01	=	4.71E+00	9.68E−01	=	4.85E+00	1.24E+00	=
F16	1.70E+01	4.47E−01	1.70E+01	4.42E−01	=	1.80E+01	8.62E−01	+	**1.60E+01**	**1.15E+00**	=
F17	1.75E+03	3.48E+02	1.41E+03	3.65E+02	−	**8.10E+02**	**3.02E+02**	−	1.18E+03	1.30E+03	−
F18	1.01E+02	1.16E+01	9.73E+01	1.38E+01	=	3.16E+01	1.02E+01	−	**8.97E−01**	**6.10E−01**	−
F19	9.30E+00	1.74E+00	8.39E+00	1.99E+00	=	1.04E+01	1.14E+00	+	**7.50E+00**	**1.86E+00**	=
F20	1.43E+01	4.85E+00	1.39E+01	4.56E+00	=	9.73E+00	3.05E+00	−	**3.09E+00**	**1.17E+00**	−
F21	6.34E+02	2.40E+02	5.15E+02	1.49E+02	−	**4.49E+02**	**1.37E+02**	−	1.45E+03	5.30E+02	+
F22	1.32E+02	7.59E+01	**1.26E+02**	**7.67E+01**	=	1.49E+02	7.62E+01	+	1.66E+02	6.39E+01	+
F23	**3.44E+02**	**3.59E−13**	**3.44E+02**	**5.74E−14**	=	**3.44E+02**	**4.59E−13**	=	3.44E+02	2.24E−05	+
F24	2.76E+02	5.73E−01	2.75E+02	6.62E−01	=	2.67E+02	4.86E+00	=	2.68E+02	1.57E+00	=
F25	**2.05E+02**	**4.38E−01**	2.05E+02	3.56E−01	+	2.05E+02	3.03E−01	=	2.17E+02	1.12E+00	+
F26	**1.00E+02**	**1.94E−02**	1.02E+02	1.40E+01	+	**1.00E+02**	**4.39E−02**	+	1.17E+02	3.07E+01	+
F27	3.29E+02	3.07E+01	3.33E+02	3.03E+01	+	3.06E+02	1.50E+01	−	**3.00E+02**	**7.02E−03**	−
F28	**1.12E+03**	**2.78E+01**	1.12E+03	2.98E+01	=	1.13E+03	4.48E+01	+	1.23E+03	5.86E+01	+
F29	7.98E+02	3.42E+01	8.12E+02	4.51E+01	+	8.30E+02	5.13E+01	+	**5.00E+02**	**2.19E+01**	−
F30	8.80E+03	4.38E+02	8.86E+03	5.03E+02	+	**8.40E+03**	**3.58E+02**	=	8.81E+03	2.70E+02	+
	+/−/=		11/14/5			9/11/10			12/5/13		

on 25, 18, 22, 30, and 23 problems out of 30 problems respectively. Performance of SS worse than CMA-ES, I-POP-CMAES, LS-CMAES, and (1+1)Cholesky-CMAES on five, nine, eight, and six problems respectively, and SS is significantly similar to the I-POP-CMAES, and (1+1)Cholesky-CMAES for three, and one problems of CEC 2014 problem suite respectively.

4. *SS vs Group-IV's algorithms:* In Tables 9 and 10, the outcomes of SS and Group-IV's algorithms are presented. The last rows of Tables 9 and 10 summarize the results of WT.

Table 15
Mean and SD of best error value obtained in 51 independent runs by SASS, iCMAES-ILS, NBIPOP-aCMA-ES, SHADE, LSHADE, EBOwithCMAR, and HSES on 100-*D* CEC2014 problem suite (Mean: Mean of best error, SD: Standard deviation of best error, W: result of Wilcoxon signed rank test).

Prob	SASS		iCMAES-ILS			NBIPOP-aCMA-ES			SHADE		
	Mean	SD	Mean	SD	W	Mean	SD	W	Mean	SD	W
F1	1.72E+05	5.67E+04	**0.00E+00**	**0.00E+00**	−	**0.00E+00**	**0.00E+00**	−	1.43E+05	1.05E+05	−
F2	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=
F3	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=	5.63E−02	2.11E−01	+
F4	9.96E+01	4.72E+01	1.14E+02	4.74E+01	+	3.74E+01	4.84E+01	−	9.25E+01	4.68E+01	=
F5	2.06E+01	3.60E−02	**2.00E+01**	1.67E−05	−	2.12E+01	3.12E−01	+	2.02E+01	1.47E−02	−
F6	1.55E+01	3.34E+00	**7.12E−02**	2.24E−01	−	5.38E+00	5.27E+00	−	3.35E+01	5.25E+00	+
F7	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=	1.98E−03	5.34E−03	+
F8	8.41E−04	4.65E−04	1.94E+01	3.05E+00	+	4.56E+01	1.39E+01	+	**0.00E+00**	**0.00E+00**	−
F9	5.81E+01	5.47E+00	2.01E+01	3.30E+00	−	**6.93E+00**	**3.17E+00**	−	1.14E+02	1.55E+01	+
F10	1.55E+01	3.73E+00	9.31E+02	6.51E+02	+	5.16E+03	2.53E+03	+	**1.03E−02**	**7.58E−03**	−
F11	1.10E+04	5.01E+02	**1.51E+03**	**7.96E+02**	−	3.72E+03	1.71E+03	−	9.80E+03	4.97E+02	−
F12	4.31E−01	4.18E−02	**6.62E−04**	**5.94E−04**	−	1.12E−02	5.16E−03	−	2.29E−01	2.44E−02	−
F13	2.55E−01	1.96E−02	2.39E−01	3.18E−02	=	2.70E−01	6.87E−02	+	4.00E−01	4.28E−02	+
F14	2.31E−01	9.32E−03	1.35E−01	1.63E−02	−	**1.18E−01**	**1.35E−02**	−	1.20E−01	9.78E−03	−
F15	1.57E+01	1.26E+00	9.79E+00	8.37E−01	−	**9.68E+00**	**1.00E+00**	−	2.21E+01	2.87E+00	+
F16	3.93E+01	4.32E−01	4.25E+01	7.68E−01	+	4.29E+01	2.00E+00	+	3.96E+01	4.93E−01	+
F17	4.43E+03	6.79E+02	5.07E+03	7.47E+02	+	4.84E+03	6.24E+02	+	1.24E+04	4.80E+03	+
F18	2.15E+02	1.26E+01	5.43E+02	1.69E+02	+	6.60E+02	9.88E+01	+	5.07E+02	3.89E+02	+
F19	9.89E+01	3.08E+00	8.21E+01	3.35E+01	=	1.02E+02	1.61E+01	+	9.78E+01	1.55E+01	=
F20	2.25E+02	5.94E+01	6.53E+02	1.24E+02	+	6.16E+02	1.22E+02	+	4.92E+02	1.34E+02	+
F21	2.24E+03	5.11E+02	3.32E+03	7.39E+02	+	3.39E+03	6.66E+02	+	3.69E+03	1.62E+03	+
F22	1.09E+03	2.14E+02	5.38E+02	1.78E+02	−	7.62E+02	2.50E+02	−	1.40E+03	2.13E+02	+
F23	**3.48E+02**	**2.49E−13**	**3.48E+02**	**5.74E−14**	=	**3.48E+02**	**5.74E−14**	=	**3.48E+02**	**5.74E−14**	=
F24	3.95E+02	2.81E+00	**3.53E+02**	**9.49E+00**	−	3.88E+02	2.61E+00	=	3.94E+02	4.45E+00	=
F25	**2.00E+02**	**4.92E−13**	2.18E+02	1.86E+00	+	2.18E+02	2.19E+00	+	2.58E+02	5.87E+00	+
F26	2.00E+02	4.59E−13	1.91E+02	2.72E+01	+	**1.49E+02**	**5.03E+01**	−	2.00E+02	1.26E−02	+
F27	5.39E+02	6.26E+01	**3.00E+02**	**1.93E−01**	−	**3.00E+02**	**2.77E+00**	−	9.50E+02	8.84E+01	+
F28	2.21E+03	6.10E+01	2.49E+03	8.67E+01	+	2.46E+03	3.05E+02	+	2.32E+03	2.51E+02	+
F29	**7.52E+02**	**4.03E+01**	1.51E+03	7.18E+02	+	7.54E+02	3.03E+01	+	1.35E+03	1.49E+02	+
F30	7.09E+03	1.04E+03	8.71E+03	1.32E+03	+	9.04E+03	1.75E+03	+	8.50E+03	9.27E+02	+
	+/−/=		13/6/11			14/10/6			18/5/7		

Prob	SS		LSHADE			EBOwithCMAR			HSES		
	Mean	SD	Mean	SD	W	Mean	SD	W	Mean	SD	W
F1	1.72E+05	5.67E+04	1.72E+05	5.65E+04	=	5.17E−03	5.36E−04	−	1.28E−01	1.03E−01	−
F2	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	=	2.50E−09	7.01E−09	+	1.71E−09	6.21E−09	+
F3	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	=	3.89E−07	5.84E−07	+	1.18E−08	4.96E−08	+
F4	9.96E+01	4.72E+01	1.70E+02	3.06E+01	+	1.55E+02	2.82E+01	+	**2.97E+01**	**5.29E+01**	−
F5	2.06E+01	3.60E−02	2.06E+01	3.11E−02	=	**2.00E+01**	**1.04E−04**	−	**2.00E+01**	**1.52E−04**	−
F6	1.55E+01	3.34E+00	8.69E+00	2.33E+00	−	7.30E−01	8.46E−01	−	1.22E+00	1.07E+00	−
F7	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=	1.99E−09	9.25E−09	+
F8	8.41E−04	4.65E−04	1.11E−02	7.44E−03	+	6.15E−01	2.79E+00	+	1.34E+01	5.96E+00	+
F9	5.81E+01	5.47E+00	3.44E+01	5.00E+00	−	2.92E+01	4.26E+00	−	8.82E+00	3.70E+00	−
F10	1.55E+01	3.73E+00	2.64E+01	5.79E+00	+	9.69E+01	1.93E+02	+	1.59E+03	1.01E+03	+
F11	1.10E+04	5.01E+02	1.08E+04	5.62E+02	=	1.09E+04	1.40E+03	=	3.03E+03	5.17E+02	−
F12	4.31E−01	4.18E−02	4.37E−01	4.65E−02	+	5.81E−02	2.85E−02	−	8.67E−04	1.04E−03	−
F13	2.55E−01	1.96E−02	2.41E−01	2.10E−02	−	2.50E−01	7.40E−02	=	**6.21E−02**	**1.18E−02**	−
F14	2.31E−01	9.32E−03	**1.18E−01**	**7.35E−03**	−	1.82E−01	1.24E−02	−	2.47E−01	2.00E−02	+
F15	1.57E+01	1.26E+00	1.62E+01	1.20E+00	+	1.16E+01	1.55E+00	−	1.18E+01	1.88E+00	=
F16	3.93E+01	4.32E−01	3.91E+01	4.80E−01	=	4.22E+01	1.29E+00	+	**3.80E+01**	**1.89E+00**	=
F17	4.43E+03	6.79E+02	4.43E+03	7.13E+02	=	3.99E+03	7.76E+02	=	**1.55E+03**	**2.11E+03**	−
F18	2.15E+02	1.26E+01	2.24E+02	1.68E+01	+	2.17E+02	2.89E+01	+	**2.79E+00**	**1.68E+00**	−
F19	9.89E+01	3.08E+00	9.57E+01	2.29E+00	=	9.01E+01	1.28E+00	−	**7.35E+01**	**2.23E+01**	=
F20	2.25E+02	5.94E+01	1.50E+02	5.25E+01	−	**5.81E+01**	**1.56E+01**	−	3.21E+02	1.94E+02	+
F21	2.24E+03	5.11E+02	2.27E+03	5.31E+02	+	**1.30E+03**	**4.18E+02**	−	3.17E+03	6.55E+02	+
F22	1.09E+03	2.14E+02	1.11E+03	1.88E+02	+	1.10E+03	3.03E+02	+	**6.90E+02**	**4.09E+02**	−
F23	**3.48E+02**	**2.49E−13**	**3.48E+02**	**3.66E−13**	=	**3.48E+02**	**5.74E−14**	+	**3.48E+02**	**3.68E−03**	+
F24	3.95E+02	2.81E+00	3.94E+02	2.87E+00	=	3.77E+02	4.77E+00	−	3.76E+02	2.26E+00	=
F25	**2.00E+02**	**4.92E−13**	**2.00E+02**	**4.09E−13**	+	2.18E+02	1.27E+00	+	2.04E+02	7.99E+00	+
F26	2.00E+02	4.59E−13	2.00E+02	6.27E−13	+	2.00E+02	1.46E−02	+	2.00E+02	3.81E−02	+
F27	5.39E+02	6.26E+01	3.77E+02	3.28E+01	−	3.01E+02	3.90E+00	−	3.00E+02	1.67E−05	−
F28	2.21E+03	6.10E+01	2.31E+03	4.65E+01	+	**2.14E+03**	**2.99E+02**	=	2.29E+03	2.41E+02	+
F29	**7.52E+02**	**4.03E+01**	7.98E+02	7.58E+01	+	9.51E+02	1.77E+02	+	7.53E+02	5.93E+01	+
F30	7.09E+03	1.04E+03	8.29E+03	9.63E+02	+	**6.51E+03**	**9.53E+02**	=	6.98E+03	6.35E+03	=
	+/−/=		13/11/6			12/6/12			13/5/12		

It is seen from Tables 9 and 10 that the performance of SS is better than GWO, GOA, MVO, SCA, SHO, SSA, SOA, and WOA on 22, 28, 21, 23, 22, 21, 21, and 23 problems out of 30 problems, respectively. The performance of SS is outperformed by GWO, GOA, MVO, SCA, SHO, SSA, SOA, and WOA on seven, zero, eight, three, six, eight, seven, and seven problems out of 30 problems , respectively.

In addition, the Friedman Ranking (FR) test is also used to detect the significant differences between SS and the other 23 algorithms on all the 30 problems of CEC 2014 problem suite. The

Table 16
Ranking of algorithm according to Friedman ranking based on mean error value. (FR: Friedman Ranking).

Algorithm	10-D		30-D		50-D		100-D		Total	
	FR	Rank	FR	Rank	FR	Rank	FR	Rank	FR	Rank
SASS	**3.1333**	**2**	**3.4333**	**1.5**	**3.8500**	**3**	**3.7000**	**2.5**	**3.5292**	**2**
iCMAES-ILS	3.9833	4	4.2000	5	3.9333	4	**3.7000**	**2.5**	3.9542	5
NBIPOP-aCMAES	4.8000	6	4.2000	5	4.3667	6	4.2167	6	4.3958	6
SHADE	4.6000	5	5.0833	7	5.1500	7	5.2500	7	5.0208	7
LSHADE	**3.9500**	**3**	**3.4500**	**3**	4.0667	5	4.1667	5	**3.9083**	**3**
EBOwithCMAR	**2.6833**	**1**	**3.4333**	**1.5**	**3.2167**	**1**	3.7833	4	**3.2792**	**1**
HSES	4.8500	7	4.2000	5	**3.4167**	**2**	**3.1833**	**1**	3.9125	4

Table 17
Comparison among the performance of SASS with the performance of other state-of-art algorithms on problem IEEE CEC 2011. Best entries are shown in bold face. (Mean: Average of error achieved for 51 independent runs, SD: standard deviation of error achieved for 51 independent runs, W: outcome of Wilcoxon signed rank test at 0.05 significance level).

Prob	SASS		GA-MPC			SHADE			L-SHADE		
	Mean	SD	Mean	SD	W	Mean	SD	W	Mean	SD	W
T1	1.30E−03	5.38E−03	9.99E−01	3.52E+00	+	5.64E−01	3.97E+00	+	1.40E−04	6.45E−04	=
T2	−2.59E+01	5.43E−01	−2.71E+01	8.43E−01	−	−2.41E+01	5.59E−01	+	−2.61E+01	6.27E−01	=
T3	**1.15E−05**	**1.73E−21**	**1.15E−05**	**5.67E−19**	=	**1.15E−05**	**1.68E−19**	=	**1.15E−05**	**2.07E−19**	=
T4	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**3.32E+00**	=	1.61E+01	3.02E+00	+
T5	−3.62E+01	7.67E−01	−3.40E+01	1.47E+00	+	**−3.65E+01**	**5.30E−01**	−	−3.62E+01	7.69E−01	=
T6	−2.91E+01	3.46E−01	−2.71E+01	2.44E+00	+	**−2.92E+01**	**4.24E−03**	=	−2.92E+01	4.78E−03	=
T7	1.16E+00	8.05E−02	2.06E+00	4.35E−02	+	**6.11E−01**	**9.03E−02**	−	1.17E+00	9.38E−02	+
T8	**2.20E+02**	**0.00E+00**	**2.20E+02**	**0.00E+00**	=	**2.20E+02**	**0.00E+00**	=	**2.20E+02**	**0.00E+00**	=
T9	2.88E+03	7.32E+02	2.27E+03	9.89E+02	−	1.98E+03	3.61E+02	−	2.65E+03	4.36E+02	−
T10	−2.16E+01	9.06E−02	−2.17E+01	1.27E−01	−	**−2.18E+01**	**9.91E−02**	=	−2.16E+01	8.70E−02	=
T11	**5.19E+04**	**3.64E+02**	5.38E+04	8.22E+03	+	5.25E+04	5.06E+02	+	5.20E+04	5.03E+02	+
T12	**1.07E+06**	**1.41E+03**	1.08E+06	1.19E+04	+	1.07E+06	4.43E+04	+	1.78E+07	5.68E+04	+
T13	**1.54E+04**	**1.30E−06**	1.54E+04	8.75E−06	=	1.54E+04	1.09E+01	+	1.54E+04	6.16E−01	+
T14	**1.81E+04**	**2.07E+01**	1.83E+04	8.50E+01	+	1.81E+04	3.37E+01	+	1.81E+04	3.34E+01	+
T15	**3.27E+04**	**9.25E−01**	3.28E+04	3.86E+01	+	3.27E+04	1.16E+01	+	3.27E+04	3.31E−01	+
T16	**1.24E+05**	**4.30E+02**	1.36E+05	2.74E+03	+	1.30E+05	4.60E+02	+	1.24E+05	5.58E+02	+
T17	1.86E+06	1.26E+04	2.17E+06	4.80E+05	+	1.90E+06	1.22E+04	+	1.87E+06	9.83E+03	+
T18	9.33E+05	1.30E+03	1.03E+06	1.65E+05	+	9.40E+05	9.82E+02	+	9.32E+05	9.63E+02	=
T19	9.41E+05	8.85E+02	1.30E+06	1.99E+05	+	9.77E+05	1.15E+03	+	9.40E+05	9.00E+02	−
T20	9.33E+05	1.29E+03	1.04E+06	4.28E+04	+	9.40E+05	1.16E+03	+	9.32E+05	1.16E+03	−
T21	1.61E+01	6.37E−01	**1.30E+01**	**3.41E+00**	−	1.86E+01	7.75E−01	+	1.62E+01	7.61E−01	+
T22	1.37E+01	2.10E+00	**1.05E+01**	**2.13E+00**	−	1.49E+01	2.42E+00	+	1.34E+01	1.90E+00	−
	+/=/−		13/4/5			14/5/3			10/8/4		

Prob	SASS		L-SHADE-EpSin			EBO			EBOwithCMAR		
	Mean	SD	Mean	SD	W	Mean	SD	W	Mean	SD	W
T1	1.30E−03	5.38E−03	9.73E−01	3.44E+00	+	1.66E−01	3.56E−01	+	**7.08E−12**	**5.26E−12**	−
T2	−2.59E+01	5.43E−01	−2.59E+01	5.81E−01	=	−2.32E+01	1.31E+00	+	**−2.71E+01**	**6.94E−01**	−
T3	**1.15E−05**	**1.73E−21**	**1.15E−05**	**1.98E−19**	=	**1.15E−05**	**4.45E−20**	=	**1.15E−05**	**1.73E−21**	=
T4	**0.00E+00**	**0.00E+00**	1.45E+01	1.08E+00	+	**0.00E+00**	**0.00E+00**	=	**0.00E+00**	**0.00E+00**	=
T5	−3.62E+01	7.67E−01	**−3.65E+01**	**5.05E−01**	=	−3.59E+01	5.42E−01	+	−3.59E+01	7.53E−01	+
T6	−2.91E+01	3.46E−01	−2.92E+01	7.77E−03	=	−2.91E+01	3.46E−01	=	−2.89E+01	1.27E+00	+
T7	1.16E+00	8.05E−02	1.15E+00	9.18E−02	=	1.31E+00	8.56E−02	+	7.36E−01	1.49E−01	−
T8	**2.20E+02**	**0.00E+00**	**2.20E+02**	**0.00E+00**	=	**2.20E+02**	**0.00E+00**	=	**2.20E+02**	**0.00E+00**	=
T9	2.88E+03	7.32E+02	1.91E+03	3.41E+02	−	**1.67E+03**	**3.54E+02**	−	2.46E+03	5.22E+02	−
T10	−2.16E+01	9.06E−02	−2.16E+01	9.40E−02	+	−2.15E+01	1.34E−01	+	−2.16E+01	1.00E−01	=
T11	**5.19E+04**	**3.64E+02**	5.21E+04	6.05E+02	+	5.23E+04	2.68E+02	+	5.20E+04	4.63E+02	+
T12	**1.07E+06**	**1.41E+03**	1.78E+07	5.86E+04	+	1.07E+06	1.61E+03	=	1.07E+06	1.69E+03	=
T13	**1.54E+04**	**1.30E−06**	1.55E+04	1.34E+01	+	1.54E+04	5.13E−07	=	1.54E+04	7.43E−12	−
T14	**1.81E+04**	**2.07E+01**	1.81E+04	4.40E+01	+	1.81E+04	1.64E+02	=	1.81E+04	1.95E+01	=
T15	**3.27E+04**	**9.25E−01**	3.27E+04	8.29E+00	+	3.27E+04	1.34E+01	+	3.27E+04	3.25E+00	+
T16	**1.24E+05**	**4.30E+02**	1.24E+05	5.31E+02	+	1.26E+05	5.87E+02	+	1.26E+05	6.18E+02	+
T17	1.86E+06	1.26E+04	**1.85E+06**	**1.00E+04**	−	1.88E+06	1.11E+04	+	1.90E+06	2.63E+04	+
T18	9.33E+05	1.30E+03	**9.30E+05**	**9.67E+02**	−	9.37E+05	1.54E+03	+	9.38E+05	1.48E+03	+
T19	9.41E+05	8.85E+02	**9.39E+05**	**8.11E+02**	−	9.44E+05	5.16E+03	+	9.47E+05	7.99E+03	+
T20	9.33E+05	1.29E+03	**9.30E+05**	**1.28E+03**	−	9.37E+05	2.55E+03	+	9.37E+05	1.82E+03	+
T21	1.61E+01	6.37E−01	1.57E+01	6.47E−01	=	1.63E+01	1.73E+00	+	1.41E+01	2.04E+00	−
T22	1.37E+01	2.10E+00	1.67E+01	2.37E+00	+	1.44E+01	1.84E+00	+	1.32E+01	3.08E+00	−
	+/=/−		10/7/5			14/7/1			9/6/7		

detailed results of the FR for all 24 algorithms are shown in Table 11. From Table 11, it can be found that SS is ranked first by FR among all 24 algorithms. Variants of PSO: BB-PSO, CLPSO, OLPSO, and variants of DE: CoBiDE, RSDE, POBL-ADE, and DE-best are very competitive with the SS but the performance of SS is slightly better than them. Similarly, variants of CMA-ES: I-POP-CMAES and LS-CMAES also perform well on the CEC-2014 problem suite, but they could not outperform the SS. Recently proposed algorithms, MVO, SOA, WOA, SSA, and GWO perform very well, but the performance of SS is significantly better than

Table 18
Ranking of algorithm according to Friedman ranking based on mean error value. (FR: Friedman Ranking).

Algorithm	Ranking	
	FR	Rank
SASS	**3.5000**	**1**
GA-MPC	5.1364	7
SHADE	4.1818	5
LSHADE	3.6591	4
LSHADEEpSin	3.5455	2
EBO	4.3636	6
EBOwithCMAR	3.6136	3

them. Compared with rest of the other algorithms, SS significantly outperformed them.

4.4. Experiment 4: SASS vs best performer of IEEE CEC's competitions

In this paper, a self-adaptation procedure is also proposed to set the value of parameters online during the optimization using past experience. The algorithm SS with self-adaptation is named as SASS. The performance of SASS is analyzed on IEEE CEC 2014 test suite. The following algorithms are chosen as a contender of SASS for comparative study.

1. NBIPOP-aCMAES: CMA-ES integrated with an occasional restart strategy and increasing population size and an iterative local search (winner of IEEE CEC'13) [49].
2. iCMAES-ILS: CMA-ES with restarts (ranked second in IEEE CEC 2013) [50].
3. SHADE: Success-history based parameter adaptation for DE (First ranked non-CMA-ES variant of IEEE CEC 2013) [31].
4. LSHADE: SHADE with Linear Population Size Reduction (winner of CEC 2014 competition) [51].
5. EBOwithCMAR: Effective Butterfly optimizer with covariance matrix adapted retreat phase (winner of IEEE CEC 2017) [52].
6. HSES: hybrid sampling evolution strategy (winner of IEEE CEC 2018) [53].

The same parameter setting is adopted for aforementioned algorithms as reported in their respective papers. Same stopping criterion, 10,000D as maximum allowed number of function evaluations, is considered for all the algorithm for 51 independent runs as suggested in [31]. In SASS, the parameters are set as follows: $N_{min} = 4$, $N_{max} = 18D$, $H = 6$, $L_1 j = 0.5$ for $j = 1$ to H and $L_{2j} = 0.5$ for $j = 1$ to H.

The mean and SD of the error values for all the algorithms, i.e. $f(x*) - f(x_{best})$, on $D = 10$, $D = 30$, $D = 50$, and $D = 100$ problems are depicted in Tables 12–15, respectively. In addition, WT at 0.05 significance level is utilized to check the significance of the performance of SASS as compared to others. In Tables 12–15, "+" sign denotes that the performance of SASS is better than the other contenders, "−" sign represents that the SASS is outperformed by other contenders, and "=" shows that the there is no significant difference in the performance of SASS as compared to other contenders. The last rows of these tables summarize the outcomes of WT in terms of +(win)/=(tie)/−(lose).

From Table 12, it is found that the performance of SASS is better than the iCMAES-ILS, NBIPOP-aCMAES, SHADE, LSHADE, EBOwithCMAR, and HSES on 15, 17, 17, 17, eight, 16 problems out of 30 problems of 10D, respectively. SASS is outperformed by iCMAES-ILS, NBIPOP-aCMAES, SHADE, LSHADE, EBOwithCMAR, and HSES on six, five, zero, three, seven, and five problems of 10D, respectively. From the statistical outcomes of 10D problems, it can be concluded that the performance of SASS is better than other contenders except for EBOwithCMAR. SASS performs similar to the EBOwithCMAR on 10D problems.

An analysis of the results of WT on 30D problems (Table 13) shows that the SASS performs better than iCMAES-ILS, NBIPOP-aCMAES, SHADE, LSHADE, EBOwithMAR, and HSES on 14, 14, 20, 11, 11, and 13 problems, respectively. The performance of iCMAES-ILS, NBIPOP-aCMAES, SHADE, LSHADE, EBOwithCMAR, and HSES is better than SASS on seven, seven, one, three, seven, and eight problems out of 30 problems of 30D, respectively. This comparative analysis reveals that the performance of SASS is comparatively better than the other algorithms on 30D problems.

In the case of 50D problems, the SASS outperforms iCMAES-ILS, NBIPOP-aCMAES, SHADE, LSHADE, EBOwithCMAR, and HSES on 13, 14, 19, 11, nine, and 12 problems as shown in Table 14. The performance of SASS is inferior to iCMAES-ILS, NBIPOP-aCMAES, SHADE, LSHADE, EBOwithCMAR, and HSES on eight, eight, six, five, 10, and 13 problems out of 30 problems of 50D. These comparative outcomes revealed that SASS exhibits competitive performance in comparison to EBOwithCMAR and HSES. Moreover, SASS provides better performance than the rest of the contenders.

For 100D problems, Table 15 shows that the SASS outperforms iCMAES-ILS, NBIPOP-aCMAES, SHADE, LSHADE, EBOwithCMAR, and HSES on 13, 14, 18, 13, 12, and 13 problems, respectively. SASS is outperformed by iCMAES-ILS, NBIPOP-aCMAES, SHADE, LSHADE, EBOwithCMAR, and HSES on 11, six, seven, six, 12, and 12 problems, respectively. From the comparative analysis, it can be concluded that the performance of SASS is competitive with HSES, EBOwithCMAR, and iCMAES-ILS and SASS outperforms NBIPOP-aCMAES, SHADE, and LSHADE on 100D problems.

In addition, each algorithm is ranked using FT. In Table 16, the outcomes of FT are reported for all the algorithms on 10D, 30D, 50D, and 100D problems of IEEE CEC 2014 suite. The obtained results show that SASS is ranked 2, 1.5, 3, and 2.5 on 10D, 30D, 50D, and 100D problems, respectively. SASS is outperformed by EBOwithCMAR on 10D problems, EBOwithCMAR and HSES on 50D problems, and HSES on 100D problems on the basis of mean rank of FT. The last column of Table 16 summarizes the overall ranking of algorithms obtained by FT. As shown in Table 16, SASS is the second best performing algorithm on IEEE CEC 2014 problem suite as compared to all contenders. EBOwithCMAR provides the best performance. From the above analysis, it can be concluded that the performance of SASS is superior or at least competitive with the best performers of IEEE CEC competitions.

4.5. Application to real-life complex optimization problems: IEEE CEC 2011 problem suite

SASS is also benchmarked on real-world optimization problems to analyze the effectiveness of SASS in solving these types of optimization problems. Optimization problems of IEEE CEC 2011 [54] problem suite are selected as test problems. This problem suite contains 22 challenging problems from different engineering fields viz. optimal power flow, economic dispatch problems, network expansion planning, radar system, spacecraft trajectory optimization, and sound waves. The dimensions of these problems vary from six to 40.

The parametric setting is same as reported in the experiment 4.4. The outcomes of SASS are compared statistically with six algorithms including the winners of IEEE CEC competitions. Each contender is employed for 25 independent runs on each problem of IEEE CEC 2011 problem suite taking 50,000 function evaluations as a stopping criterion as suggested in [54]. The following six algorithms are considered in this experiment.

1. GA-MPC: GA with a new Multi-Parent Crossover (winner of the IEEE CEC 2011 competition) [55].
2. SHADE: Success-History based parameter adaptation for Differential Evolution [31].
3. LSHADE: SHADE with Linear population size reduction (winner of CEC 2014 competition) [51] .
4. LSHADEEpSin: LSHADE with an ensemble Sinusoidal parameter adaptation (winner of CEC 2016 competetition) [56].
5. EBO: Effective Butterfly Optimizer [52].
6. EBOwithCMAR: EBO with Covariance Matrix Adapted Retreat phase (winner of CEC 2017 competition) [52].

A comparison of SASS with the aforementioned contenders is performed and reported in Table 17. In Table 17, mean and standard deviation (SD) of the best objective function values of each run for each algorithm is shown. The Wilcoxon's signed rank test (W) is also utilized to check the significance of the outcomes of SASS with a comparison of other algorithms. In Table 17, the last row summarizes the outcome of WT in terms of a number of superior, equal, and inferior performance ('+', '=', and '−' respectively) of SASS. The results of Table 17 show that the SASS outperforms GA-MPC, SHADE, LSHADE, LSHADEEpSin, EBO, and EBOwithCMAR on 13, 14, 10, 10, 14, and nine problems out of 22 problems, respectively. However, SASS is also outperformed by GA-MPC, SHADE, LSHADE, LSHADEEpSin, EBO, and EBOwithCMAR on five, three, four, five, one, and seven problems out of 22 problems, respectively, nevertheless this clearly proves that the performance of SASS is better than the other aforementioned contenders over IEEE CEC 2011 problem suite. Moreover, a Friedman Ranking (FR) is also done and depicted in Table 18 for each algorithm. The SASS shows best mean rank equal to 3.5 followed by EBOwithCMAR. From the results of statistical tests and comparative analysis of the performance of all algorithms, it can be concluded that SASS is an effective and robust optimization technique for real-world problems.

4.6. Discussions

In the context of this work, it can be concluded that the SS algorithm is free from the tuning of parameters that depends upon the characteristics of optimization problems. The performance of this algorithm highly depends upon the projection matrix and search direction. Projection matrix improves the diversity of solutions in population, while it requires more extra computational burden. Two approaches are used to calculate search direction which provide a good balance between exploration and exploitation. However, these approaches are somewhat influenced by step-control parameter. In order to address this issue, a parameter adaptation is also proposed to tune the parameters online during the optimization process. Thus, by analyzing the outcomes of detailed experiments and statistical tests on two popular advanced benchmark suites, it can be concluded that the performance of the proposed algorithm is better or highly competitive with the advanced variant of state-of-the-art algorithms. However, the performance of SASS (SS with parameter adaptation) is outperformed by the advanced variant of state-of-art algorithms in a few cases.

5. Conclusion

This paper proposed a new optimization technique, Spherical Search algorithm. The proposed method was tested on modern unconstrained benchmark problems to analyze the performance. The experimental results show that SS algorithm provides superior result on most of the benchmark problems compared to state-of-the-art metaheuristics. The results obtained from all experiments and comparison of characteristics with other algorithms conclude the following.

1. SS has few parameters to tune, and it is easy to implement for solving unconstrained optimization problems.
2. The quality and accuracy of obtained solutions, the rate of convergence, proficiency, and effectiveness of SS are at an advanced degree as compared to its opponents.
3. SS possibly escapes from the situation of stagnation and benefits to get rid of sticking in local minima, due to its projection property.

In the future, it is proposed to develop versions of SS for constrained, binary, and multi-objective optimization problems.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.asoc.2019.105734.

References

[1] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1) (1997) 67–82.
[2] K.S. Tang, K.F. Man, S. Kwong, Q. He, Genetic algorithms and their applications, IEEE Signal Process. Mag. 13 (6) (1996) 22–37.
[3] I. Rechenberg, Evolution strategy: Nature's way of optimization, in: Optimization: Methods and Applications, Possibilities and Limitations, Springer, 1989, pp. 106–126.
[4] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Trans. Evol. Comput. 3 (2) (1999) 82–102.
[5] L.J. Fogel, A.J. Owens, M.J. Walsh, Artificial Intelligence Through Simulated Evolution, John Wiley, 1966.
[6] J.R. Koza, Genetic Programming: On the Programming of Computers By Means of Natural Selection, Vol. 1, MIT press, 1992.
[7] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (4) (1997) 341–359.
[8] J. Kennedy, Particle swarm optimization, in: Encyclopedia of Machine Learning, Springer, 2010, pp. 760–766.
[9] M. Dorigo, M. Birattari, Ant colony optimization, in: Encyclopedia of Machine Learning, Springer, 2010, pp. 36–39.
[10] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, J. Global Optim. 39 (3) (2007) 459–471.
[11] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer, Adv. Eng. Softw. 69 (2014) 46–61.
[12] X.-S. Yang, A new metaheuristic bat-inspired algorithm, in: Nature Inspired Cooperative Strategies for Optimization, NICSO 2010, Springer, 2010, pp. 65–74.
[13] G. Dhiman, V. Kumar, Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems, Knowl.-Based Syst. 165 (2019) 169–196.
[14] S. Mirjalili, A. Lewis, The whale optimization algorithm, Adv. Eng. Softw. 95 (2016) 51–67.
[15] S. Mirjalili, The ant lion optimizer, Adv. Eng. Softw. 83 (2015) 80–98.
[16] M. Yazdani, F. Jolai, Lion optimization algorithm (LOA): a nature-inspired metaheuristic algorithm, J. Comput. Design Eng. 3 (1) (2016) 24–36.
[17] S. Balochian, H. Baloochian, Social mimic optimization algorithm and engineering applications, Expert Syst. Appl. (2019).
[18] A. Kaveh, A. Dadras, A novel meta-heuristic optimization algorithm: thermal exchange optimization, Adv. Eng. Softw. 110 (2017) 69–84.
[19] H. Rakhshani, A. Rahati, Snap-drift cuckoo search: A novel cuckoo search optimization algorithm, Appl. Soft Comput. 52 (2017) 771–794.

[20] M.D. Li, H. Zhao, X.W. Weng, T. Han, A novel nature-inspired algorithm for optimization: Virus colony search, Adv. Eng. Softw. 92 (2016) 65–88.
[21] S. Saremi, S. Mirjalili, A. Lewis, Grasshopper optimisation algorithm: theory and application, Adv. Eng. Softw. 105 (2017) 30–47.
[22] S. Mirjalili, A.H. Gandomi, S.Z. Mirjalili, S. Saremi, H. Faris, S.M. Mirjalili, Salp swarm algorithm: A bio-inspired optimizer for engineering design problems, Adv. Eng. Softw. 114 (2017) 163–191.
[23] M. Jain, V. Singh, A. Rani, A novel nature-inspired algorithm for optimization: Squirrel search algorithm, Swarm Evol. Comput. 44 (2019) 148–175.
[24] A. Mortazavi, V. Toğan, A. Nuhoğlu, Interactive search algorithm: a new hybrid metaheuristic optimization algorithm, Eng. Appl. Artif. Intell. 71 (2018) 275–292.
[25] G. Dhiman, V. Kumar, Emperor penguin optimizer: A bio-inspired algorithm for engineering problems, Knowl.-Based Syst. 159 (2018) 20–50.
[26] J. Pierezan, L.D.S. Coelho, Coyote optimization algorithm: a new metaheuristic for global optimization problems, in: 2018 IEEE Congress on Evolutionary Computation, CEC, IEEE, 2018, pp. 1–8.
[27] G. Dhiman, V. Kumar, Spotted hyena optimizer: a novel bio-inspired based metaheuristic technique for engineering applications, Adv. Eng. Softw. 114 (2017) 48–70.
[28] J.D. Ser, E. Osaba, D. Molina, X.-S. Yang, S. Salcedo-Sanz, D. Camacho, S. Das, P.N. Suganthan, C.A.C. Coello, F. Herrera, Bio-inspired computation: Where we stand and what's next, Swarm Evol. Comput. 48 (2019) 220–250.
[29] M. Locatelli, A note on the griewank test function, J. Global Optim. 25 (2) (2003) 169–174.
[30] A. Kumar, R.K. Misra, D. Singh, S. Das, Testing a multi-operator based differential evolution algorithm on the 100-digit challenge for single objective numerical optimization, in: Evolutionary Computation, CEC, 2019 IEEE Congress on, IEEE, 2019, pp. 34–40.
[31] R. Tanabe, A. Fukunaga, Success-history based parameter adaptation for differential evolution, in: Evolutionary Computation, CEC, 2013 IEEE Congress on, IEEE, 2013, pp. 71–78.
[32] J. Liang, B. Qu, P. Suganthan, Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore, 2013.
[33] J. Kennedy, Bare bones particle swarms, in: Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE, IEEE, 2003, pp. 80–87.
[34] K. Mahadevan, P. Kannan, Comprehensive learning particle swarm optimization for reactive power dispatch, Appl. Soft Comput. 10 (2) (2010) 641–652.
[35] Z.-H. Zhan, J. Zhang, Y. Li, H.S.-H. Chung, Adaptive particle swarm optimization, IEEE Trans. Syst. Man Cybern. B 39 (6) (2009) 1362–1381.
[36] Z.-H. Zhan, J. Zhang, Y. Li, Y.-H. Shi, Orthogonal learning particle swarm optimization, IEEE Trans. Evol. Comput. 15 (6) (2011) 832–847.
[37] Y. Li, J. Feng, J. Hu, Covariance and crossover matrix guided differential evolution for global numerical optimization, SpringerPlus 5 (1) (2016) 1176.
[38] Z. Li, Z. Shang, B.Y. Qu, J.-J. Liang, Differential evolution strategy based on the constraint of fitness values classification, in: Evolutionary Computation, CEC, 2014 IEEE Congress on, IEEE, 2014, pp. 1454–1460.
[39] C. Xu, H. Huang, S. Ye, A differential evolution with replacement strategy for real-parameter numerical optimization, in: Evolutionary Computation, CEC, 2014 IEEE Congress on, IEEE, 2014, pp. 1617–1624.
[40] Z. Hu, Y. Bao, T. Xiong, Partial opposition-based adaptive differential evolution algorithms: evaluation on the CEC 2014 benchmark set for real-parameter optimization, in: Evolutionary Computation, CEC, 2014 IEEE Congress on, IEEE, 2014, pp. 2259–2265.
[41] N. Hansen, The CMA evolution strategy: a comparing review, in: Towards a New Evolutionary Computation, Springer, 2006, pp. 75–102.
[42] A. Auger, N. Hansen, A restart CMA evolution strategy with increasing population size, in: Evolutionary Computation, 2005. the 2005 IEEE Congress on, Vol. 2, IEEE, 2005, pp. 1769–1776.
[43] A. Auger, M. Schoenauer, N. Vanhaecke, LS-CMA-ES: A second-order algorithm for covariance matrix adaptation, in: International Conference on Parallel Problem Solving from Nature, Springer, 2004, pp. 182–191.
[44] G.A. Jastrebski, D.V. Arnold, Improving evolution strategies through active covariance matrix adaptation, in: Evolutionary Computation, 2006. CEC 2006. IEEE Congress on, IEEE, 2006, pp. 2814–2821.
[45] C. Igel, T. Suttorp, N. Hansen, A computational efficient covariance matrix update and a (1+1)-CMA for evolution strategies, in: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, ACM, 2006, pp. 453–460.
[46] S. Mirjalili, S.M. Mirjalili, A. Hatamlou, Multi-verse optimizer: a nature-inspired algorithm for global optimization, Neural Comput. Appl. 27 (2) (2016) 495–513.
[47] S. Mirjalili, SCA: a sine cosine algorithm for solving optimization problems, Knowl.-Based Syst. 96 (2016) 120–133.
[48] E. Cuevas, F. Fausto, A. González, The selfish herd optimizer, in: New Advancements in Swarm Algorithms: Operators and Applications, Springer, 2020, pp. 69–109.
[49] T. Liao, T. Stuetzle, Benchmark results for a simple hybrid algorithm on the CEC 2013 benchmark set for real-parameter optimization, in: 2013 IEEE Congress on Evolutionary Computation, IEEE, 2013, pp. 1938–1944.
[50] I. Loshchilov, CMA-ES with restarts for solving CEC 2013 benchmark problems, in: 2013 IEEE Congress on Evolutionary Computation, IEEE, 2013, pp. 369–376.
[51] R. Tanabe, A.S. Fukunaga, Improving the search performance of SHADE using linear population size reduction, in: 2014 IEEE Congress on Evolutionary Computation, CEC, IEEE, 2014, pp. 1658–1665.
[52] A. Kumar, R.K. Misra, D. Singh, Improving the local search capability of effective butterfly optimizer using covariance matrix adapted retreat phase, in: 2017 IEEE Congress on Evolutionary Computation, CEC, IEEE, 2017, pp. 1835–1842.
[53] G. Zhang, Y. Shi, Hybrid sampling evolution strategy for solving single objective bound constrained problems, in: 2018 IEEE Congress on Evolutionary Computation, CEC, IEEE, 2018, pp. 1–7.
[54] S. Das, P.N. Suganthan, Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems, Jadavpur University, Nanyang Technological University, Kolkata, 2010, pp. 341–359.
[55] S.M. Elsayed, R.A. Sarker, D.L. Essam, Ga with a new multi-parent crossover for solving ieee-cec2011 competition problems, in: 2011 IEEE Congress of Evolutionary Computation, CEC, IEEE, 2011, pp. 1034–1040.
[56] N.H. Awad, M.Z. Ali, P.N. Suganthan, R.G. Reynolds, An ensemble sinusoidal parameter adaptation incorporated with L-SHADE for solving cec2014 benchmark problems, in: 2016 IEEE Congress on Evolutionary Computation, CEC, IEEE, 2016, pp. 2958–2965.