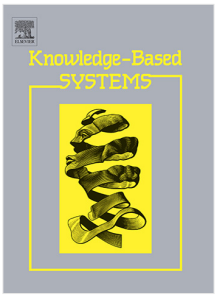
# Accepted Manuscript




Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems

Gaurav Dhiman, Vijay Kumar

PII: S0950-7051(18)30576-8
DOI: https://doi.org/10.1016/j.knosys.2018.11.024
Reference: KNOSYS 4587

To appear in: *Knowledge-Based Systems*

Received date : 11 May 2018
Revised date : 15 November 2018
Accepted date : 18 November 2018

Please cite this article as: G. Dhiman and V. Kumar, Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems, *Knowledge-Based Systems* (2018), https://doi.org/10.1016/j.knosys.2018.11.024

# Seagull Optimization Algorithm: Theory and its Applications for Large Scale Industrial Engineering Problems

Gaurav Dhiman[a*], Vijay Kumar[b]

[a*,b]Computer Science and Engineering Department, Thapar Institute of Engineering and Technology, Patiala, Punjab, INDIA
*Corresponding author email: gdhiman0001@gmail.com, gaurav.dhiman@thapar.edu

**Abstract**

This paper presents a novel bio-inspired algorithm called Seagull Optimization Algorithm (SOA) for solving computationally expensive problems. The main inspiration of this algorithm is the migration and attacking behaviors of a seagull in nature. These behaviors are mathematically modeled and implemented to emphasize exploration and exploitation in a given search space. The performance of SOA algorithm is compared with nine well-known metaheuristics on forty-four benchmark test functions. The analysis of computational complexity and convergence behaviors of the proposed algorithm have been evaluated. It is then employed to solve seven constrained real-life industrial applications to demonstrate its applicability. Experimental results reveal that the proposed algorithm is able to solve challenging large scale constrained problems and is very competitive algorithm as compared with other optimization algorithms.

*Keywords:* Optimization; Bio-inspired Metaheuristics; Industrial Problems; Benchmark Test Problems.

## 1. Introduction

Optimization is a process of determining the decision variables of a function to minimize or maximize its values. Most of the real world problems have non-linear constraints, high computational cost, non-convex, complicated, and large number of solution spaces [1]. Therefore, solving such problems with large number of variables and constraints is very tedious and complex [2–6]. Secondly, there are many local optimum solutions that do not guarantee the best solution using classical numerical methods.

To overcome these problems, metaheuristic optimization algorithms are introduced which are capable of solving such complex problems [7, 8]. Recently, there is a lot of interest in developing metaheuristic algorithms that are computationally inexpensive, flexible, and simple by nature [9–12].

Metaheuristics are broadly classified into two categories such as single solution and population based algorithms [13]. Single solution based algorithms are in which a solution is randomly generated and improved until the best result is obtained. Population based algorithms are in which a set of solutions is randomly generated in a given search space and solution values are updated during course of iterations until the best solution is found.

However, single solution based algorithms may trap into local optima that preventing us to find global optimum. This is because it reforms only one solution which is randomly generated for a given problem. On the other hand, population based algorithms are able to find the global optimum. Due to this, researchers have attracted towards population based algorithms nowadays.

The categorization of population based algorithms is done that is based on the theory of evolutionary algorithms, logical behavior of physics algorithms, swarm intelligence of particles, and biological behavior of bio-inspired algorithms. Evolutionary algorithms are inspired by the evolutionary processes such as reproduction, mutation, recombination, and selection. These algorithms are based on the survival fitness of candidate in a population (i.e., a set of solutions) for a given environment [14]. The physics law based algorithms are inspired by physical processes according to some physics rules such as gravitational force, electromagnetic force, inertia force, heating, and cooling of materials. Swarm intelligence based algorithms are inspired by the collective intelligence of swarms. The collective intelligence is found among colonies of flocks, ants, and so on.

Generally, swarm intelligence based algorithms are easy to implement than the evolutionary algorithms due to less number of parameters required. The well-known swarm intelligence based algorithms are Particle Swarm Optimization (PSO) [15] and Ant Colony Optimization [16]. The reason behind the popularity of these algorithms is that only few parameters are required for fine tuning.

Every optimization algorithm needs to address the exploration and exploitation of a search space [17] and maintains a good balance between exploration and exploitation. The exploration phase in an algorithm investigates the different promising regions in a search space whereas exploitation is able to search the optimal solutions around the promising regions [18]. Therefore, there is a need for fine tuning of these two phases to achieve the near optimal solutions. Despite the significant number of recently developed optimization algorithms, the question is raised why we need to develop more optimization techniques. The answer lies in No Free Lunch (NFL) theorem [19]. According to this theorem, the performance of one optimization algorithm for a specific set of problems does not guarantee to solve other optimization problems because of their different nature. NFL theorem allows researchers to propose some novel optimization algorithms for solving the problems in different fields [20]. This motivated us to develop a novel technique which is gradient free and effective optimization algorithm to optimize real-life engineering problems.

This paper presents a novel bio-inspired metaheuristic algorithm named as Seagull Optimization Algorithm (SOA) for optimizing the constrained problems. As its name implies, SOA mimics the migration and attacking behaviors of seagulls in nature. The performance of SOA algorithm is evaluated on forty-four well-known

benchmark test functions. Further, SOA algorithm is employed to optimize the designs of optical buffer, pressure vessel, speed reducer, welded beam, tension/compression spring, 25-bar truss, and rolling element bearing problems.

The rest of this paper is structured as follows: Section 2 presents the preliminaries of optimization and related work done in the field of single-objective optimization. Section 3 presents the proposed SOA algorithm in detail. The experimental results and discussion is presented in Section 4. In Section 5, the performance of SOA is tested on seven real-life constrained industrial optimization problems and compared it with other well-known algorithms. Finally, the conclusion and some future research directions are discussed in Section 6.

## 2. Background

This section first describes the definitions of single objective optimization. Further, the literature review and motivation of proposed work are discussed.

### 2.1. Single-objective optimization problems

Many real-world problems need to achieve several objectives such as minimize risks, maximize reliability, minimize cost, etc. The main goal of optimization process is to find the best optimal solution that combines different objectives into one. In the optimization process, there is a set of equality and inequality constraints which is to be minimized or maximized.

$$\text{Minimize/Maximize: } F(\vec{z}) = f_1(\vec{z}) \tag{1}$$

Subject to:

$$g_j(\vec{z}) \geq 0,\ j = 1, 2, \ldots, p \tag{2}$$

$$h_j(\vec{z}) = 0,\ j = 1, 2, \ldots, q \tag{3}$$

$$lb_j \leq z_j \leq ub_j,\ j = 1, 2, \ldots, r \tag{4}$$

where $p$ is the number of inequality constraints, $q$ is the number of equality constraints, $r$ is the number of variables, $lb_j$ is the lower bound, and $ub_j$ is the upper bound of the $j^{th}$ variable.

Therefore, the presence of local solutions when solving optimization problems is the main challenging task because there is only one best solution in the search space. However, there are many other solutions which are closer to the obtained objective value. These kinds of solutions can be considered as best local solutions, but not the best globally while considering the entire search space. This may trapped in local optimum due to the presence of local solutions. The convergence speed is another issue because an algorithm that avoids local solutions does not necessarily converge to global optimum. The above-mentioned are the main challenges of metaheuristic algorithm that solves real-life problems.

### 2.2. Literature review

Population based metaheuristic algorithms are broadly classified into three categories namely Physics-based, Evolutionary-based, and Swarm-based methods.

The first approach is Physics-based algorithms in which search agents can communicate around the search space according to laws of physics such as electromagnetic force, inertia force, gravitational force. The general mechanism of these algorithms is different from other approaches due to the strategy of search agents as per physics rules. The popular Physics-based optimization algorithms are Simulated Annealing (SA) [21] and Gravitational Search Algorithm (GSA) [22]. Simulated Annealing is inspired from annealing in metallurgy that involves heating and controlled cooling attributes of a material. These attributes depend on its thermodynamic free energy. Gravitational Search Algorithm is based on the law of gravity and mass interactions. The population solutions are interact with each other through the gravity force and their performance is measured by its mass. Some of the other popular algorithms are: Big-Bang Big-Crunch (BBBC) [23], Charged System Search (CSS) [24], Black Hole (BH) [25] algorithm, Central Force Optimization (CFO) [26], Small-World Optimization Algorithm (SWOA) [27], and Ray Optimization (RO) algorithm [28].

The second subclass of metaheuristic algorithms is Evolutionary-based algorithms. These algorithms are inspired by theory of natural selection and biological evolution. These algorithms often perform well to find near optimal solutions because they do not make any belief about the basic fitness landscape. The most popular evolutionary algorithm is Genetic Algorithm (GA). GA was first proposed by Holland [29]. The evolution starts with the randomly generated individuals from a population. The fitness value of each individual is calculated in each generation. The crossover and mutation operators are applied on the individuals to create a new population. Thus, the best individuals can generate the new population due to their higher probability during the course of iterations. Differential Evolution (DE) [30] is another Evolutionary-based algorithm which optimize a problem by maintaining a candidate solutions and creates new candidate solutions by combining the existing ones. It keeps the candidate solution which has best fitness value for optimization problem. Apart from these, some of the other popular evolutionary-based algorithms are Genetic Programming (GP) [31], Evolution Strategy (ES) [32], and Biogeography-Based Optimizer (BBO) [33].

The third main branch in metaheuristics is Swarm-based algorithms which are based on the collective behavior of social creatures. These techniques are generally inspired from natural colonies, flock, and herds. The most popular algorithm is Particle Swarm Optimization (PSO) which was first proposed by Kennedy and Eberhart [15]. In PSO, particles move around the search space using a combination of best solution which they have individually found and the best solution that any particle in their neighbourhood has found [58]. The whole process is repeated until the termination criterion is satisfied. Ant Colony Optimization (ACO) is another popular swarm intelligence algorithm which was proposed by Dorigo [16]. The main inspiration behind this algorithm is the social behaviour of ants in ant colony. The social intelligence of ants is able to find the shortest path between the source food and nest. Bat-inspired Algorithm (BA) [59] is another intelligence based algorithm that is inspired by the echolocation behaviour of bats. The another well-known swarm based technique is Artificial Bee Colony (ABC) algorithm [60] which is inspired by the col-

Table 1: Metaheuristic Optimization Algorithms.

| Algorithms | Abbre iation |
|---|---|
| Emperor Penguin Optimizer [34] | EPC |
| Collective Neurodynamic Optimization [1, 35–37] | CNO |
| Artificial Chemical Reaction Optimization Algorithm [38] | ACROA |
| Galaxy-based Search Algorithm [39] | GbS A |
| Firefly Algorithm [40] | FA |
| Exchange Market Algorithm [41] | EMA |
| Social-Based Algorithm [42] | SBA |
| Harmony Search [43] | HS |
| Grey Wolf Optimizer [44] | GWO |
| Mine Blast Algorithm [45] | MBA |
| Monkey Search [46] | MS |
| Bacterial Foraging Optimization Algorithm [47] | BFOA |
| Fruit fly Optimization Algorithm [48] | FOA |
| Bird Mating Optimizer [49] | BMO |
| Krill Herd [50] | KH |
| Artificial Fish-Swarm Algorithm [51] | AFSA |
| Dolphin Partner Optimization [52] | DPO |
| Wolf Pack Search Algorithm [53] | WPSA |
| Bee Collecting Pollen Algorithm [54] | BCPA |
| Termite Algorithm [55] | TA |
| Wasp Swarm Algorithm [56] | WSA |
| Marriage in Honey Bees Optimization Algorithm [57] | MBO |

lective behaviour of bees to find the food sources. Spotted Hyena Optimizer (SHO) [61–63] is a recently developed bio-inspired optimization algorithm that mimics the searching, hunting, and attacking behaviors of spotted hyenas in nature. The main concept of this technique is the social relationship and collective behavior of spotted hyenas for hunting strategy. Cuckoo Search (CS) [64] is another bio-inspired optimization approach which is inspired by the obligate brood parasitism of cuckoo species. These species laying their eggs in the nest of other species. Each egg and a cuckoo egg represent a solution and a new solution, respectively. There are also some of the other metaheuristic techniques which are tabulated in Table 1.

According to the best of our knowledge, there is no Swarm-based technique that mimics the migration and attacking behaviours of seagulls. This motivates us to propose a new Swarm-based technique which is inspired by seagulls and implement its mathematical model for solving various benchmark and real engineering problems.

## 3. Seagull optimization algorithm (SOA)

In this section, the inspiration and mathematical modeling of proposed algorithm is discussed in detail.

### *3.1. Biological paradigm*

Seagulls, scientific named as Laridae, are sea birds which can be found all over the planet. There is wide range of seagulls species with different masses and lengths. Seagulls are omnivorous and eat insects, fish, reptiles, amphibians, earthworms, and so on. Body of most seagulls is covered with white plumage. Seagulls are very intelligent birds. They use bread crumbs to attract fish and produce rain-like sound with their feet to attract earthworms hidden under the ground. Seagulls can drink both fresh and salt water. Most of animals are unable to do this. However, seagulls have a special pair of glands right above their eyes which is specifically designed to flush the salt from their systems through openings in the bill.

Generally, seagulls live in colonies. They use their intelligence to find and attack the prey. The most important thing about the seagulls is their migrating and attacking behaviors. Migration is defined as the seasonal movement of seagulls from one place to another to find the richest and most abundant food sources that will provide adequate energy [65]. This behavior is described as follows:

- During migration, they travel in a group. The initial positions of seagulls are different to avoid the collisions between each other.
- In a group, seagulls can travel towards the direction of best survival fittest seagull, i.e., a seagull whose fitness value[1] is low as compared to others.
- Based on the fittest seagull, other seagulls can update their initial positions.

Seagulls frequently attack migrating birds over the sea [66] when they migrate from one place to another. They can make their spiral natural shape movement during attacking. A conceptual model of these behaviors is illustrated in Fig. 1. These behaviors can be formulated in such a way that it can be associated with the objective function to be optimized. This makes it possible to formulate a new optimization algorithm. This paper focuses two natural behaviors of seagulls.

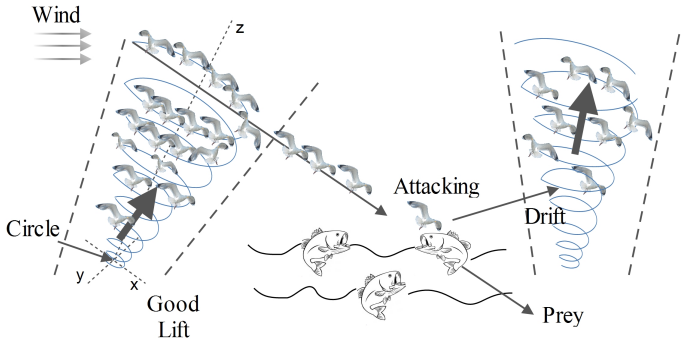



Figure 1: Migration and attacking behaviors of seagulls.

### 3.2. Mathematical model

The mathematical models of migration and attacking the prey are discussed.

#### 3.2.1. Migration (exploration)

During migration, the algorithm simulates how the group of seagulls move towards one position to another. In this phase, a seagull should satisfy three conditions:

- **Avoiding the collisions:** To avoid the collision between neighbours (i.e., other seagulls), an additional variable $A$ is employed for the calculation of new search agent position (see Fig. 2).

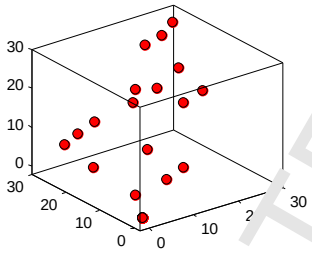


Figure 2: Collision avoidance between search agents.

$$\vec{C_s} = A \times \vec{P_s}(x) \tag{5}$$

where $\vec{C_s}$ represents the position of search agent which does not collide with other search agent, $\vec{P_s}$ represents the current position of search agent, $x$ indicates the current iteration, and $A$ represents the movement behavior of search agent in a given search space.

$$A = f_c - (x \times (f_c / Max_{iteration}))$$
$$\text{where: } x = 0, 1, 2, \ldots, Max_{iteration} \tag{6}$$

where $f_c$ is introduced to control the frequency of employing variable $A$ which is linearly decreased from $f_c$ to 0. In this work, the value of $f_c$ is set to 2. The detailed sensitivity analysis of $f_c$ has been discussed in Section 4.9.

[1]The term fitness value is defined as a process which evaluates the population and gives a score or fitness. Whereas, the process is a function which measures the quality of the represented solution.

- **Movement towards best neighbor's direction:** After avoiding the collision between neighbours, the search agents are move towards the direction of best neighbour (see Fig. 3).

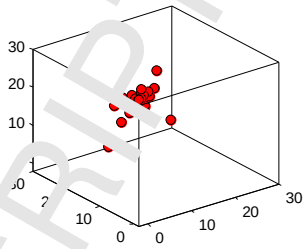


Figure 3: Movement of search agents towards the best neighbour.

$$\vec{M_s} = B \times (\vec{P_{bs}}(x) - \vec{P_s}(x)) \tag{7}$$

where $\vec{M_s}$ represents the positions of search agent $\vec{P_s}$ towards the best fit search agent $\vec{P_{bs}}$ (i.e., fittest seagull). The behavior of $B$ is randomized which is responsible for proper balancing between exploration and exploitation. $B$ is calculated as:

$$B = 2 \times A^2 \times rd \tag{8}$$

where $rd$ is a random number lies in the range of [0, 1].

- **Remain close to the best search agent:** Lastly, the search agent can update its position with respect to best search agent which is shown in Fig. 4.

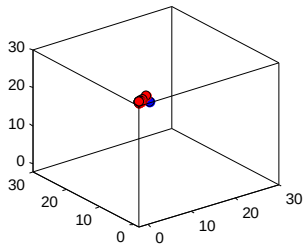


Figure 4: Convergence towards the best search agent.

$$\vec{D_s} = | \vec{C_s} + \vec{M_s} | \tag{9}$$

where $\vec{D_s}$ represents the distance between the search agent and best fit search agent (i.e., best seagull whose fitness value is less).

#### 3.2.2. Attacking (exploitation)

The exploitation intends to exploit the history and experience of the search process. Seagulls can change the angle of attack continuously as well as speed during migration. They maintain their altitude using their wings and weight. While attacking the prey, the spiral movement behavior occurs in the air (see Fig. 5). This behavior in $x, y$, and $z$ planes is described as follows.

$$x' = r \times cos(k) \tag{10}$$

$$y' = r \times sin(k) \tag{11}$$

$$z' = r \times k \tag{12}$$

$$r = u \times e^{kv} \tag{13}$$

where $r$ is the radius of each turn of the spiral, $k$ is a random number in range $[0 \leq k \leq 2\pi]$. $u$ and $v$ are constants to define the spiral shape, and $e$ is the base of the natural logarithm. The updated position of search agent is calculated using Eqs. (9) - (13).

$$\vec{P_s}(x) = (\vec{D_s} \times x' \times y' \times z') + \vec{P_{bs}}(x) \tag{14}$$

where $\vec{P_s}(x)$ saves the best solution and updates the position of other search agents.

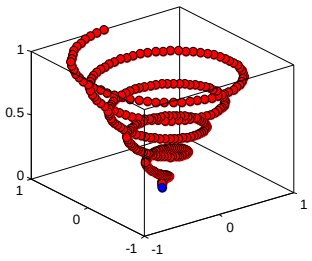


Figure 5: Natural attacking behavior of seagull.

The proposed SOA starts with a random generated population. The search agents can update their positions with respect to best search agent during the iteration process. $A$ is linearly decreased from $f_c$ to 0. For smooth transition between exploration and exploitation, variable $B$ is responsible. Hence, SOA is considered as a global optimizer (see Algorithm 1) because of its better exploration and exploitation capability.

### 3.3. Computational complexity

The complexity of an algorithm is an important metric to judge its performance. The population initialization process of proposed SOA and other competitor algorithms (i.e., SHO, GWO, PSO, MFO, MVO, SCA, GSA, GA, and DE) requires $O(n_o \times n_p)$ time, where $n_o$ represents the number of objectives and $n_p$ represents the number of population size. The complexity of calculating the fitness of search agents for all algorithms needs $O(Max_{iteration} \times o_f)$, where $o_f$ represents the objective function for a given problem. For simulate the whole procedure, it requires $O(N)$ time.

The computational complexity of proposed SOA algorithm is $O(N \times Max_{iteration} \times n_o \times n_p \times o_f)$. The computational complexity of SHO algorithm is $O(N \times G \times Max_{iteration} \times n_o \times n_p \times o_f)$ since it requires $O(G)$ time to define the group of spotted hyenas. The computational complexities corresponding to GWO, PSO, MFO, MVO, SCA algorithms are $O(N \times Max_{iteration} \times n_o \times n_p \times o_f)$. The GSA algorithm needs $O(f^2)$ time for calculating the force component and overall requires $O(N \times f^2 \times Max_{iteration} \times n_o \times n_p \times o_f)$. Finally, the computational complexities of GA, and DE algorithms are $O(N \times Max_{iteration} \times n_o \times n_p \times o_f \times (c_s + m_t))$, where $c_s$ and $m_t$ represent the crossover and mutation operators, respectively.

Whereas, the space complexity of all algorithms are the maximum amount of utilized space at any one time which is considered during its initialization process. In this work, the space complexity for all approaches are considered as $O(n_o \times n_p)$.

However, the average running time of proposed SOA and other algorithms is given in Table 2 and Fig. 6. It can be seen that SOA takes less time than other approaches in terms of seconds.

Since, SOA is a bio-inspired optimization algorithm so there is no need of evolutionary processes, i.e., crossover and mutation. Therefore, it can be concluded that the computational efficiency of proposed algorithm is much better than the other competitor approaches.

Table 2: Average running time of proposed and competitive approaches.

| Algorithms | Average time (*In seconds*) |
|---|---|
| Seagull Optimization Algorithm (SOA) | 1.2245 |
| Spotted Hyena Optimizer (SHO) | 1.3269 |
| Grey Wolf Optimizer (GWO) | 1.2536 |
| Particle Swarm Optimization (PSO) | 2.0241 |
| Moth-Flame Optimization (MFO) | 2.2362 |
| Multi-Verse Optimizer (MVO) | 1.5236 |
| Sine Cosine Algorithm (SCA) | 1.4452 |
| Gravitational Search Algorithm (GSA) | 1.2965 |
| Genetic Algorithm (GA) | 1.3668 |
| Differential Evolution (DE) | 1.6354 |

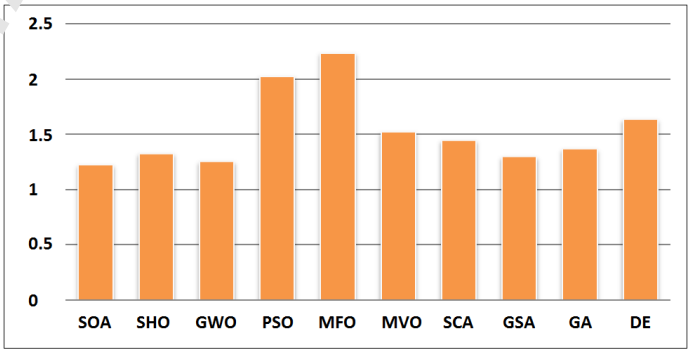



Figure 6: Average running time of metaheuristics algorithms.

## 4. Experimental results and discussion

In this section, the proposed algorithm is tested on 44 well-known benchmark test functions (see Appendix A) and the results are compared with the other competitor methods. These test functions are categorised into five categories such as unimodal, multimodal, fixed-dimension multimodal, CEC 2005 [61], and CEC 2015 special session test functions [67]. To demonstrate the efficiency of SOA algorithm, seven real-life constrained industrial optimization applications are also employed.

To validate the performance of the proposed algorithm, the nine well-known metaheuristics are chosen for comparison. These are Spotted Hyena Optimizer (SHO) [61], Grey Wolf Optimizer (GWO) [44], Particle Swarm Optimization (PSO) [15], Moth-Flame Optimization (MFO) [68], Multi-Verse Optimizer (MVO) [69], Sine Cosine Algorithm (SCA) [70], Gravitational Search Algorithm (GSA) [22], Genetic Algorithm (GA) [71], and Differential Evolution (DE) [30].

**Algorithm 1** Seagull Optimization Algorithm

**Input:** Seagull population $\vec{P_s}$
**Output:** Optimal search agent $\vec{P_{bs}}$

**procedure** SOA
Initialize the parameters $A$, $B$, and $Max_{iteration}$
    Set $f_c \leftarrow 2$
    Set $u \leftarrow 1$
    Set $v \leftarrow 1$
    **while** $(x < Max_{iteration})$ **do**
        $\vec{P_{bs}} \leftarrow$**ComputeFitness**$(\vec{P_s})$ /* Calculate the fitness values of each search agent using **ComputeFitness** function*/
    /* Migration behavior */
        $rd \leftarrow Rand(0, 1)$ /* To generate the random number in range [0, 1] */
        $k \leftarrow Rand(0, 2\pi)$ /* To generate the random number in range [0, $2\pi$] */
    /* Attacking behavior */
        $r \leftarrow u \times e^{kv}$ /* To generate the spiral behavior during migration */
        Calculate the distance $\vec{D_s}$ using Eq. (9)
        $P \leftarrow x' \times y' \times z'$ /* Compute x, y, z planes using Eqs. (10) - (13) */
        $\vec{P_s}(x) \leftarrow (\vec{D_s} \times P) + \vec{P_{bs}}$
        $x \leftarrow x + 1$
    **end while**
return $\vec{P_{bs}}$
**end procedure**

**procedure** ComputeFitness($\vec{P_s}$)
    **for** i $\leftarrow$ 1 to n **do** /* Here, n represents the dimension of a given problem */
        $FIT_s[i] \leftarrow FitnessFunction(P_s(i,:))$ /* Calculate the fitness of each individual */
    **end for**
    $FIT_{s_{best}} \leftarrow$ **BEST**$(FIT_s[])$ /* Calculate the best fitness value using **BEST** function */
return $FIT_{s_{best}}$
**end procedure**

**procedure** BEST($FIT_s[]$)
    $Best \leftarrow FIT_s[0]$
    **for** i $\leftarrow$ 1 to n **do**
        **if**$(FIT_s[i] < Best)$ **then**
            $Best \leftarrow FIT_s[i]$
        **end if**
    **end for**
return $Best$ /* Return the best fitness value */
**end procedure**

### 4.1. Experimental setup

The parameter settings of SOA and other competitor algorithms are given in Table 3. The experimentation and algorithms are implemented in MATLAB R2018b (version 9.5.0) software. The simulations are performed in the environment of Microsoft Windows 8.1 (64 bit) on Core i5 processor with 3.2 GHz and 8 GB main memory. The proposed algorithm utilizes 30 independent runs to generate the results.

Table 3: Parameter setting values for algorithms.

| Algorithms | Parameters | Values |
|---|---|---|
| *# For all algorithms* | Population | 100 |
| | Maximum iterations | 1000 |
| Seagull Optimization Algorithm (SOA) | Control Parameter (A) | [2, 0] |
| | $f_c$ | 2 |
| Spotted Hyena Optimizer (SHO) | Control Parameter ($\vec{h}$) | [5, 0] |
| | $\vec{M}$ Constant | [0.5, 1] |
| Grey Wolf Optimizer (GWO) | Control Parameter ($\vec{a}$) | [2, 0] |
| Particle Swarm Optimization (PSO) | Inertia Coefficient | 0.75 |
| | Cognitive and Social Coeff | 1.8, 2 |
| Moth-Flame Optimization (MFO) | Convergence Constant | [-1, -2] |
| | Logarithmic Spiral | 0.75 |
| Multi-Verse Optimizer (MVO) | Wormhole Existence Prob. | [0.2, 1] |
| | Travelling Distance Rate | [0.6, 1] |
| Sine Cosine Algorithm (SCA) | Number of Elites | 2 |
| Gravitational Search Algorithm (GSA) | Gravitational Constant | 1 |
| | Alpha Coefficient | 20 |
| Genetic Algorithm (GA) | Crossover | 0.9 |
| | Mutation | 0.5 |
| Differential Evolution (DE) | Crossover | 0.9 |
| | Scale factor ($F$) | 0.5 |

### 4.2. Exploitation analysis

The unimodal test functions are suitable to assess and evaluate the exploitation capability of the algorithm. Table 4 shows the optimal values obtained by the proposed and above-mentioned algorithms for unimodal benchmark test functions. It can be seen that SOA and SHO algorithms provide better results on $F_1(Sphere)$ test function. GWO and GSA are the second and third best algorithms in terms of average and standard deviation. For $F_2(Schwefel's\ Problem\ 2.22)$ test function, SOA and SHO algorithms are significantly better than the other competitor algorithms. After these, GWO and GSA are the second and third best algorithms for this benchmark test function, respectively. SHO algorithm obtains the best results on $F_3(Schwefel's\ Problem\ 1.2)$ test function whereas SOA and GWO algorithms become second and third best optimizer, respectively. For $F_4(Schwefel's\ Problem\ 2.21)$ test function, GWO obtains better results than the other algorithms. SHO is the second best and SOA is the third best optimizer on this benchmark test function. The results obtained by SOA algorithm are better than the other approaches on $F_5(Generalized\ Rosenbrock's)$ test function. For $F_5$ test function, SHO and GWO are the second and third best optimization algorithm. GSA algorithm obtains competitive results than the other metaheuristics on $F_6(Step)$ test function. On this benchmark test function, PSO and MFO algorithms are the second and third most best optimization algorithms. For $F_7(Quartic)$ test function, the results obtained by SOA algorithm are superior than others. Whereas, SHO and GA algorithms outperform over other algorithms as a second and third best algorithm, respectively. Therefore, it has been concluded that the proposed SOA algorithm is able to find the best results on most of the test functions. In particular, SOA reveals the robustness for functions $F_1$, $F_2$, $F_5$, and $F_7$ which show the capability for exploitation around the optimum point.

### 4.3. Exploration analysis

These functions have a capability to avoid local optima problem. Multimodal and fixed-dimension multimodal test functions are suitable for better exploration of a proposed algorithm. Tables 5 and 6 depict the results of different methods on multimodal and fixed-dimension multimodal benchmark test functions, respectively. For $F_8(Generalized\ Schwefel's\ Problem\ 2.26)$ function, SOA algorithm outperforms than the existing competitive approaches. MFO and MVO are the second and third best optimization algorithms, respectively. For $F_9(Generalized\ Rastrigin's\ Function)$ function, SHO becomes the best optimizer than others. GWO and GA are the second and third best optimization algorithms on $F_9$ test function. For $F_{10}(Ackley's\ Function)$ function, the average value of SOA algorithm is best as compared to other approaches whereas the standard deviation of GWO algorithm is better than the other metaheuristics. For $F_{11}(Generalized\ Griewank\ Function)$ function, the performance of SOA and SHO algorithms are same whereas the performance of GA and DE algorithms are the second and third best than competitive optimizers. The results obtained by PSO algorithm are superior than other metaheuristics on $F_{12}(Generalized\ Penalized\ Function)$ function. For $F_{13}(Generalized\ Penalized\ Function)$ function, GSA is the best optimization algorithm while PSO and GA obtain better results than the others.

For $F_{14}(Shekel's\ Foxholes\ Function)$ function, MVO provides better results. SCA and MFO are second and third best optimization algorithms, respectively. For $F_{15}(Kowalik's\ Function)$ and $F_{16}(Six-Hump\ Camel-Back\ Function)$ functions, the results of SOA are superior than the other algorithms. DE and SHO algorithms are the second and third best optimizers, respectively. For $F_{17}(Branin\ Function)$ and $F_{18}(Goldstein-Price\ Function)$ functions, the results produced from SOA algorithm are better than others. For $F_{19}(Hartman's)$ function, the average value obtained by PSO algorithm is better while the standard deviation value obtained by MFO algorithm is better than the other competitive approaches. For $F_{20}(Hartman's)$ function, SOA algorithm performs superior whereas PSO and GWO are the second and third best optimizers. The average value of GWO algorithm is best on $F_{21}(Shekel's\ Foxholes\ Function)$ test function while the standard deviation value of SHO algorithm is better on this test instance. The results obtained by SOA algorithm are superior than the competitor approaches on $F_{22}(Shekel's\ Foxholes\ Function)$ and $F_{23}(Shekel's\ Foxholes\ Function)$ benchmark test functions. It can be seen that SOA algorithm has good exploration capability and competitive on seven test problems (i.e., $F15, F16, F17, F18, F20, F22$, and $F23$) out of ten test functions.

### 4.4. Escape local minima with CEC 2005 test functions

Table 7 depicts the results obtained by all algorithms in terms of average and standard deviation. For $CF1$ test function, GSA algorithm outperforms than the other competitor metaheuristics.

For $CF2$ test function, the results of SOA are superior than others whereas MFO and SCA algorithms are second and third best optimizers, respectively. For $CF3$ test function, the performance of SOA algorithm is better than the above-mentioned algorithms. The performance of SHO and PSO are very much similar on this test function. For $CF4$ test function, the results obtained by MFO algorithm are better than the existing metaheuristics. For $CF5$ and $CF6$ test functions, SOA algorithm outperforms the other optimization algorithms. SHO and MFO algorithms are the second and third most best optimizers on $CF5$ and $CF6$ test functions. It has been concluded that the proposed SOA algorithm provides prominent results for the majority of test functions (i.e., $CF2, CF3, CF5$, and $CF6$).

### *4.5. Global minima finding with CEC 2015 test functions*

For proper exploration and exploitation, the proposed algorithm is tested on fifteen CEC 2015 special session benchmark test functions. Table 8 shows the best obtained optimal solutions by all algorithms in terms of average and standard deviation. For $CEC-1$ test function, the average value of SOA algorithm is better than other approaches. The standard deviation value of PSO algorithm is superior on this test instance. For $CEC-2$ test function, the performance of GSA algorithm is better as compared to other metaheuristics. The average value obtained by SOA algorithm on $CEC-3$ test instance is better while the standard deviation value of GSA algorithm is best than other approaches. For $CEC-4, CEC-5$, and $CEC-6$ test functions, PSO generates optimal solutions and very competitive than other optimizers. The optimal values obtained by proposed SOA algorithm in terms of average is superior than all of the other optimization algorithms on $CEC-7, CEC-8, CEC-9, CEC-10, CEC-11, CEC-12, CEC-13, CEC-14$, and $CEC-15$ benchmark test instances. Despite, the standard deviation values of PSO algorithm on $CEC-8, CEC-9, CEC-10$, and $CEC-15$ test functions are better than others. The standard deviation obtained by GSA algorithm on $CEC-14$ test instance is superior as compared to other metaheuristic techniques. Fig. 11 shows the box plot for the proposed SOA and other competitor algorithms on CEC 2015 benchmark test functions. The results reveal that the SOA algorithm is the best optimizer for most of the test functions.

### *4.6. Analysis of SOA algorithm*

The convergence analysis of metaheuristic algorithm is another feature for better understanding of explorative and exploitative mechanisms. Seagulls tend to explore the different promising regions of the search space. During the initial generation of optimization process, the search agents can change rapidly. Fig. 8 shows the convergence curves of SOA and other competitor methods. While optimizing the test functions, SOA shows the three different convergence behaviors. In the initial step of iterations, SOA converges more rapidly towards the promising regions because of its adaptive mechanism. This behavior is shown in $F1, F3, F7, F9$ and $F11$ test functions. GSA encourages good convergence and achieves better performance on $F5$ test function duringthe initial steps of iterations. Whereas, MFO converges towards the optimum for $F21$ and $F23$ test functions. In second step, SOA converges towards the optimum during the final iteration which is observed in $F21$ and $F23$ test functions. GA and MFO algorithms are also converge on $F1$, $F3$ and $F11$ test functions, SCA and MVO on $F5$ test function, and PSO algorithm on $F7$ and $F9$ test functions towards the optimum during final iterations. The last step is the explicit convergence. This behavior is shown for SOA algorithm on $F21$ and $F23$ test functions, for MFO, GA, and DE on $F9$ test function.

In order to demonstrate the convergence analysis of SOA algorithm, three metrics are also employed in 2D environment which are shown in Fig. 7. The employed metrics are discussed as follows:

- **Search history** shows the location history of seagulls during optimization. It is observed from Fig. 7 that SOA explore most promising area in the given search space for benchmark test functions. For unimodal test functions, the sample points are sparsely distributed in the non-promising area. Whereas, the most of sample points are distributed around the promising area for multimodal and fixed-dimension multimodal test functions. This is due to the difficulty level of these test functions. SOA does not stuck in local optima and explores the entire search space. The distribution of sample points is around the true optimal solution, which ensures its exploitation capability. Therefore, SOA has both exploration and exploitation capability.

- **Trajectory** of the first seagull shows the value of the first variable in each iteration. The trajectory curves show that the seagulls exhibit large and abrupt changes in the initial steps of optimization. According to Berg *et al.* [72], this behavior can guarantees that a Swarm-based method eventually converges to a point in search space.

- **Average fitness** indicates the average objective value of all seagulls in each iteration. The curves show descending behavior on all of the test functions. This proves that SOA algorithm improves the accuracy of the approximated optimum during simulation runs.

Therefore, the success rate of SOA algorithm is computationally high for solving optimization problems.

### *4.7. Scalability study*

The proposed algorithm is performed on highly scalable environment to solve the large scale optimization problems. The dimensionality of the various test functions is varied from 30 to 50, 50 to 80, and 80 to 100. In Fig. 9, the performance of SOA algorithm provides different behaviors which shows the robustness of proposed algorithm. It is also observed that the performance degradation is not too much that making its applicability on the highly scalable environment.

### *4.8. Statistical testing*

The comparison of algorithms does not guarantee the efficacy and superiority of the proposed algorithm. The possibility of getting good results, by chance, can not be ignored. For this, Wilcoxon statistical test [73] is performed at 5% level of significance and the $p-$values are tabulated in Table 9. During statistical testing, the best algorithm is chosen and compared with other competitor algorithms.

Whereas, $p-$values are less than 0.05 to demonstrate the statistical superiority of proposed SOA. The test functions are taken from CEC 2015 special session which are most challenging test functions in the literature. Besides, the Mann-Whitney $U$ rank sum test [74] has also been conducted on the average values of CEC 2015 special session test functions. Table 10 shows the results of the Mann-Whitney $U$ rank sum test for statistical superiority of proposed SOA

Table 4: Results of SOA on unimodal benchmark test functions.

| F | SOA | | SHO | | GWO | | PSO | | MFO | | MVO | | SCA | | GSA | | GA | | DE | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | *Ave* | *Std* | *Ave* | *Std* | *Ave* | *Std* | *Ave* | *Std* | *Ave* | *Std* | *Ave* | *Std* | *Ave* | *Std* | *Ave* | *Std* | *Ave* | *Std* | *Ave* | *Std* |
| $F_1$ | **0.00E+00** | **0.00E+00** | 0.00E+00 | 0.00E+00 | 4.69E-47 | 4.50E-48 | 4.98E-09 | 1.41E-10 | 3.15E-04 | 5.00E-05 | 2.81E-01 | 1.10E-02 | 3.55E-02 | 1.07E-02 | 1.16E-16 | 6.10E-17 | 1.95E-12 | 2.04E-13 | 3.10E-10 | 3.21E-11 |
| $F_2$ | **0.00E+00** | **0.00E+00** | 0.00E+00 | 0.00E+00 | 1.20E-24 | 1.00E-24 | 7.29E-04 | 1.57E-05 | 3.71E+01 | 2.16E+01 | 3.96E-01 | 1.41E-01 | 3.23E-05 | 8.28E-06 | 1.70E-01 | 5.29E-02 | 6.53E-18 | 4.00E-19 | 4.99E-13 | 4.81E-14 |
| $F_3$ | 4.62E-19 | 2.30E-27 | **0.00E+00** | **0.00E+00** | 1.00E-14 | 4.18E-15 | 1.40E+01 | 7.13E+00 | 4.42E+03 | 3.71E+03 | 4.31E+01 | 8.97E+00 | 4.91E+03 | 3.89E+03 | 4.16E+02 | 1.56E+02 | 7.70E-10 | 5.46E-12 | 5.58E-02 | 1.50E-03 |
| $F_4$ | 7.35E-05 | 3.11E-06 | 78E-12 | 5.06E-12 | **2.02E-14** | **2.95E-15** | 6.00E-01 | 1.72E-01 | 6.70E+01 | 1.06E+01 | 8.80E-01 | 2.50E-01 | 1.87E+01 | 8.21E+00 | 1.12E+00 | 9.89E-01 | 9.17E+01 | 5.67E+01 | 5.21E+01 | 1.01E-01 |
| $F_5$ | **7.00E+00** | **1.13[illegible]01** | .59E+00 | 5.53E-01 | 2.79E+01 | 1.84E+00 | 4.93E+01 | 3.89E+01 | 3.50E+03 | 2.83E+03 | 1.18E+02 | 1.00E+02 | 7.37E+02 | 1.90E+02 | 3.85E+01 | 3.47E+01 | 5.57E+02 | 4.16E+01 | 2.10E+02 | 1.73E+02 |
| $F_6$ | 3.47E-02 | 1.[illegible]1E-04 | 2.46[illegible]01 | 1.78[illegible]-01 | 6.58E-01 | 3.38E-01 | 9.23E-09 | 8.48E-10 | 1.66E-04 | 2.49E-05 | 3.15E-01 | 9.98E-02 | 4.88E+00 | 9.75E-01 | **1.08E-16** | **4.00E-17** | 3.15E-01 | 9.98E-02 | 9.77E-02 | 5.13E-02 |
| $F_7$ | **3.35E-06** | **7.51E-06** | 3.2[illegible]E-05 | 2.43E-05 | [illegible]80E-04 | 3.85E-04 | 6.92E-02 | 2.87E-02 | 3.22E-01 | 2.93E-01 | 2.02E-02 | 7.43E-03 | 3.88E-02 | 1.15E-02 | 7.68E-01 | 5.70E-01 | 6.79E-04 | 5.21E-04 | 4.00E-03 | 2.27E-03 |

Table 5: Results of SOA on multimodal benchmark test functions.

| F | SOA | | SHO | | GWO | | PSO | | MFO | | MVO | | SCA | | GSA | | GA | | DE | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | *Ave* | *Std* | *Ave* | *Std* | *Ave* | *Std* | *Ave* | *Std* | *Ave* | *Std* | *Ave* | *Std* | *Ave* | *Std* | *Ave* | *Std* | *Ave* | *Std* | *Ave* | *Std* |
| $F_8$ | **-8.50E+03** | 4.57E+02 | -1.16E+03 | **2.72E+02** | -6.14E+03 | 9.32E+02 | -6.01E+03 | [illegible]30E+03 | 8.04E+03 | 8.80E+02 | -6.92E+03 | 9.19E+02 | -3.81E+03 | 2.83E+02 | -2.75E+03 | 5.72E+02 | -5.11E+03 | 4.37E+02 | -3.94E+03 | 5.81E+02 |
| $F_9$ | 3.12E+02 | 2.85E+01 | **0.00E+00** | **0.00E+00** | 4.34E-01 | 2.68E-01 | 4.72E+01 | 1.03E+0[illegible] | 1.63[illegible]02 | 3.74E+01 | 1.01E+02 | 1.89E+01 | 2.23E+01 | 1.28E+01 | 3.35E+01 | 1.19E+01 | 1.23E+01 | 1.01E+01 | 8.50E+01 | 1.00E+01 |
| $F_{10}$ | **4.22E-16** | 4.09E-13 | 2.48E+00 | 1.41E+00 | 1.63E-14 | **3.14E-15** | 3.86E-02 | 2.91E-4 | 1.60E+01 | [illegible]18E+00 | 1.15E+00 | 7.87E-01 | 1.55E+01 | 8.11E+00 | 8.25E-09 | 1.90E-09 | 5.31E-11 | 9.10E-12 | 7.40E-06 | 5.44E-07 |
| $F_{11}$ | **0.00E+00** | **0.00E+00** | 0.00E+00 | 0.00E+00 | 2.29E-03 | 1.21E-03 | 5.50E-03 | 4.32[illegible]04 | 5.03E-0[illegible] | 1.29E[illegible]02 | 5.74E-01 | 1.12E-01 | 3.01E-01 | 2.89E-01 | 8.19E+00 | 3.70E+00 | 3.31E-06 | 1.43E-06 | 7.09E-04 | 5.58E-04 |
| $F_{12}$ | 5.80E-01 | 4.71E-02 | 3.68E-02 | 1.15E-02 | 3.93E-02 | 2.42E-02 | **1.05E-10** | **2.86E-11** | 1.26[illegible]00 | 1.83[illegible]00 | 1.27E+00 | 1.02E+00 | 5.21E+01 | 2.19E+01 | 2.65E-01 | 1.13E-01 | 9.16E-08 | 2.80E-08 | 1.06E-06 | 1.00E-06 |
| $F_{13}$ | 8.48E-02 | 4.32E-04 | 9.29E-01 | 9.52E-02 | 4.75E-01 | 2.38E-01 | 4.03E-03 | 3.01E-03 | 7.24E-[illegible] | 1.5[illegible]E-01 | 6.6[illegible]02 | 4.33E-02 | 2.81E+02 | 5.62E+01 | **5.73E-32** | **1.30E-32** | 6.39E-02 | 4.49E-02 | 9.78E-02 | 3.42E-02 |

Table 6: Results of SOA on fixed-dimension multimodal benchmark test functions.

| F | SOA | | SHO | | GWO | | PSO | | MFO | | MVO | | SCA | | GSA | | GA | | DE | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | *Ave* | *Std* | *Ave* | *Std* | *Ave* | *Std* | *Ave* | *Std* | *Ave* | *Std* | *Ave* | *Std* | *Ave* | *Std* | *Ave* | *Std* | *Ave* | *Std* | *Ave* | *Std* |
| $F_{14}$ | 3.35E+00 | 1.15E+00 | 9.68E+00 | 3.29E+00 | 3.71E+00 | 2.15E+00 | 2.77E+00 | 2.20E+00 | 2.21E+00 | 1.80E+00 | **9.98E-01** | **4.13E-02** | 1.26E+00 | .86E[illegible] | 3.61[illegible]00 | 2.96E+00 | 4.39E+00 | 4.41E-02 | 3.97E+00 | 1.42E+00 |
| $F_{15}$ | **4.11E-04** | **4.26E-05** | 9.01E-04 | 1.06E-04 | 3.66E-03 | 3.48E-04 | 9.09E-04 | 2.38E-04 | 1.58E-03 | 2.48E-04 | 7.15E-03 | 1.40E-03 | 1.01E-03 | 3.75[illegible]04 | 6.[illegible]E-03 | 2.7[illegible]03 | 7.36E-03 | 2.39E-04 | 5.01E-04 | 1.00E-04 |
| $F_{16}$ | **-1.08E+01** | **6.48E-12** | -1.06E+01 | 2.86E-11 | -1.03E+00 | 7.02E-09 | -1.03E+00 | 0.00E+00 | -1.03E+00 | 0.00E+00 | -1.03E+00 | 4.74E-08 | -1.03E+00 | 3.23E[illegible]5 | [illegible].03E+0[illegible] | 0.00[illegible]00 | -1.04E+00 | 4.19E-07 | -1.03E+00 | 3.60E-04 |
| $F_{17}$ | **3.98E-01** | 1.36E-03 | 3.98E-01 | 2.46E-01 | 3.98E-01 | 2.42E-02 | 3.98E-01 | 4.11E-02 | 3.98E-01 | 1.13E-03 | 3.98E-01 | 2.15E-02 | 3.99E-01 | 7.61E-04 | 3.98[illegible]01 | 1.[illegible]4E-04 | 3.98[illegible]01 | **1.00E-04** | 3.98E-01 | 8.40E-02 |
| $F_{18}$ | **3.00E+00** | 5.49E-05 | 3.00E+00 | 8.15E-05 | 3.00E+00 | 7.16E-06 | 3.00E+00 | 6.59E-05 | 3.00E+00 | **1.42E-07** | 5.70E+00 | 8.40E-04 | 3.00E+00 | 2.25E-05 | 3.01E[illegible]0 | [illegible].4E-[illegible]2 | 3.01E[illegible]00 | [illegible]33E-07 | 3.00E+00 | 9.68E-04 |
| $F_{19}$ | -3.88E+00 | 3.00E-10 | -3.75E+00 | 4.39E-01 | -3.86E+00 | 1.57E-03 | **-3.90E+00** | 3.37E-15 | -3.86E+00 | **3.16E-15** | -3.86E+00 | 3.53E-07 | -3.86E+00 | 2.55E-03 | -3.22E+00 | 4.15E[illegible]01 | -3.30E+0[illegible] | 4.37E[illegible]0 | -3.40E+00 | 6.56E-06 |
| $F_{20}$ | **-3.32E+00** | **1.23E-02** | -1.44E+00 | 5.47E-01 | -3.27E+00 | 7.27E-02 | -3.32E+00 | 2.66E-01 | -3.23E+00 | 6.65E-02 | -3.23E+00 | 5.37E-02 | -2.84E+00 | 3.71E-01 | -1.47E+00 | 5.32E-0[illegible] | [illegible].39E-[illegible]0 | [illegible].37E-[illegible] | [illegible].79E+00 | 7.07E-02 |
| $F_{21}$ | -1.00E+01 | 3.47E+00 | -1.00E+01 | **3.80E-01** | **-1.01E+01** | 1.54E+00 | -1.00E+01 | 2.77E+00 | -1.00E+01 | 3.52E+00 | -1.00E+01 | 2.91E+00 | -1.00E+01 | 1.80E+00 | -1.00E+01 | 1.30E+00 | -1.00[illegible]+01 | 2.3[illegible]3+00 | -1.00[illegible] | 1.91E+00 |
| $F_{22}$ | **-1.04E+01** | **2.00E-04** | -1.04E+01 | 2.04E-04 | -1.03E+01 | 2.73E-04 | -1.04E+01 | 3.08E+00 | -1.04E+01 | 3.20E+00 | -1.01E+01 | 3.02E+00 | -1.04E+01 | 1.99E+00 | -1.04E+01 | 2.64E+00 | -1.04E+01 | 1[illegible]E-02 | -1.[illegible]E+01 | [illegible].0E-03 |
| $F_{23}$ | **-1.05E+01** | **1.32E-01** | -1.05E+01 | 2.64E-01 | -1.01E+01 | 8.17E+00 | -1.05E+01 | 2.52E+00 | -1.05E+01 | 3.68E+00 | -1.03E+01 | 3.13E+00 | -1.05E+01 | 1.96E+00 | -1.05E+01 | 2.75E+00 | -1.05E+01 | 2.37E-[illegible]0 | [illegible].05E-[illegible] | [illegible].60E+00 |

Table 7: Results of SOA on CEC 2005 special session benchmark test functions.

| F | SOA | | SHO | | GWO | | PSO | | MFO | | MVO | | SCA | | GSA | | GA | | DE | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | *Ave* | *Std* | *Ave* | *Std* | *Ave* | *Std* | *Ave* | *Std* | *Ave* | *Std* | *Ave* | *Std* | *Ave* | *Std* | *Ave* | *Std* | *Ave* | *Std* | *Ave* | *Std* |
| *CF1* | 2.52E+01 | **2.11E+00** | 2.30E+02 | 1.37E+02 | 8.39E+01 | 6.02E+01 | 6.00E+01 | 5.05E+01 | 1.18E+02 | 7.40E+01 | 1.40E+02 | 1.00E+02 | 1.20E+02 | 3.11E+01 | **1.81E+01** | 1.05E+01 | 5.97E+02 | 1.34E+02 | 4.41E+02 | 3.57E+02 |
| *CF2* | **6.21E+01** | **1.30E+00** | 4.08E+02 | 9.36E+01 | 1.48E+02 | 3.78E+01 | 2.44E+02 | 1.73E+02 | 9.20E+01 | 1.84E+01 | 2.50E+02 | 1.44E+02 | 1.14E+02 | 1.84E+00 | 2.03E+02 | 1.99E+01 | 4.09E+02 | 2.10E+01 | 6.01E+02 | 1.14E+02 |
| *CF3* | **3.01E+02** | **2.35E+01** | 3.39E+02 | 3.14E+01 | 3.53E+02 | 5.88E+01 | 3.39E+02 | 8.36E+01 | 4.19E+02 | 1.15E+02 | 4.05E+02 | 1.67E+02 | 3.89E+02 | 5.41E+01 | 3.67E+02 | 8.38E+01 | 9.30E+02 | 8.31E+01 | 3.70E+02 | 2.11E+02 |
| *CF4* | 3.35E+02 | 2.63E+01 | 7.26E+02 | 1.21E+02 | 4.23E+02 | 1.14E+02 | 4.49E+02 | 1.42E+02 | **3.31E+02** | **2.09E+01** | 3.77E+02 | 1.28E+02 | 4.31E+02 | 2.94E+01 | 5.32E+02 | 1.01E+02 | 4.97E+02 | 3.24E+01 | 5.00E+02 | 2.12E+02 |
| *CF5* | **5.61E+01** | **1.15E+01** | 1.06E+02 | 1.38E+01 | 1.36E+02 | 2.75E+01 | 2.40E+02 | 7.49E+01 | 1.13E+02 | 9.27E+01 | 2.45E+02 | 9.96E+01 | 1.56E+02 | 8.30E+01 | 1.44E+02 | 1.09E+02 | 1.90E+02 | 5.03E+01 | 4.06E+02 | 3.45E+01 |
| *CF6* | **4.55E+02** | **3.73E+00** | 5.97E+02 | 4.98E+00 | 8.26E+02 | 1.74E+02 | 8.22E+02 | 1.80E+02 | 8.92E+02 | 2.41E+01 | 8.33E+02 | 1.68E+02 | 6.06E+02 | 1.66E+02 | 8.13E+02 | 1.13E+02 | 6.65E+02 | 3.37E+02 | 9.55E+02 | 6.04E+02 |

Table 8: Results of SOA on CEC 2015 benchmark test functions.

| F | SOA | SHO | GWO | PSO | MFO | MVO | SCA | GSA | GA | DE |
|---|---|---|---|---|---|---|---|---|---|---|
| *CEC-1* | **2.55E+05** | 2.28E+06 | 2.02E+06 | 4.37E+05 | 1.47E+06 | 6.06E+05 | 7.65E+06 | 3.20E+07 | 8.89E+06 | 6.09E+06 |
| | *(2.45E+05)* | *(2.18E+06)* | *(2.01E+06)* | ***(1.81E+05)*** | *(1.00E+06)* | *(5.02E+05)* | *(3.07E+06)* | *(2.98E+06)* | *(6.95E+06)* | *(5.11E+06)* |
| *CEC-2* | 5.53E+06 | 3.13E+05 | 5.65E+06 | 9.41E+03 | 1.97E+04 | 1.43E+04 | 7.33E+08 | **4.58E+03** | 2.[illegible]05 | 4.40E+04 |
| | *(8.37E+04)* | *(2.10E+05)* | *(2.19E+06)* | *(4.82E+03)* | *(1.46E+04)* | *(1.03E+04)* | *(2.33E+08)* | ***(1.09E[illegible]*** | *(2.85E+03)* | *(2.75E+04)* |
| *CEC-3* | **3.20E+02** | 3.20E+02 | 3.20E+02 | 3.20E+02 | 3.20E+02 | 3.20E+02 | 3.20E+02 | 3.20[illegible]02 | 3.20E+02 | 3.20E+02 |
| | *(3.71E-03)* | *(3.76E-02)* | *(7.08E-02)* | *(8.61E-02)* | *(9.14E-02)* | *(3.19E-02)* | *(7.53E-02)* | ***(1.[illegible]03)*** | *(2.78E-02)* | *(1.15E-03)* |
| *CEC-4* | 4.10E+02 | 4.11E+02 | 4.16E+02 | **4.09E+02** | 4.26E+02 | 4.18E+02 | 4.42E+02 | 4.[illegible]39E+02 | 6.99E+02 | 4.91E+06 |
| | *(5.32E+01)* | *(1.71E+01)* | *(1.03E+01)* | ***(3.96E+00)*** | *(1.17E+01)* | *(1.03E+01)* | *(7.72E+00)* | *(7.2[illegible]00)* | *(6.43E+00)* | *(6.03E+00)* |
| *CEC-5* | 9.54E+02 | 9.13E+02 | 9.20E+02 | **8.65E+02** | 1.33E+03 | 1.09E+03 | 1.76E+03 | [illegible]75E+03 | 1.26E+03 | 3.34E+03 |
| | *(2.12E+02)* | *(1.85E+02)* | ***(1.78E+02)*** | *(2.16E+02)* | *(3.45E+02)* | *(2.81E+02)* | *(2.30E+02)* | *([illegible])* | *(1.86E+02)* | *(3.01E+02)* |
| *CEC-6* | 2.47E+03 | 1.29E+04 | 2.26E+04 | **1.86E+03** | 7.35E+03 | 3.82E+03 | 2.30E+04 | [illegible]91E+06 | 2.91E+05 | 5.39E+04 |
| | *(1.40E+03)* | *(1.15E+04)* | *(2.07E+04)* | ***(1.28E+03)*** | *(3.82E+03)* | *(2.44E+03)* | *(1.91E+04)* | [illegible]70E+06) | *(1.67E+05)* | *(2.40E+04)* |
| *CEC-7* | **7.02E+02** | 7.02E+02 | 7.02E+02 | 7.02E+02 | 7.02E+02 | 7.02E+02 | 7.06[illegible]02 | 7.08[illegible]02 | 7.08E+02 | 7.06E+02 |
| | ***(5.76E-01)*** | *(6.76E-01)* | *(7.07E-01)* | *(7.75E-01)* | *(1.10E+00)* | *(9.40E-01)* | *(9.[illegible]E-01)* | [illegible]32E+00) | *(2.97E+00)* | *(3.87E+00)* |
| *CEC-8* | **1.65E+03** | 1.86E+03 | 3.49E+03 | 3.43E+03 | 9.93E+03 | 2.58E+03 | 6.73[illegible]03 | [illegible]07E+05 | 5.79E+04 | 5.60E+03 |
| | *(2.12E+03)* | *(1.98E+03)* | *(2.04E+03)* | *(2.77E+03)* | *(8.74E+03)* | ***(1.61E+03)*** | [illegible]36E+0[illegible] | *(4.81E+05)* | *(2.76E+04)* | *(5.15E+04)* |
| *CEC-9* | **1.00E+03** | 1.00E+03 | 1.00E+03 | 1.00E+03 | 1.00E+03 | 1.00E+03 | 1.00[illegible] | 1.00E+03 | 1.00E+03 | 1.00E+03 |
| | *(7.35E-01)* | *(1.43E-01)* | *(1.28E-01)* | *(7.23E-02)* | *(2.20E-01)* | ***(5.29E-02)*** | [illegible]9E-01) | *(5.33E+00)* | *(3.97E+00)* | *(4.05E+00)* |
| *CEC-10* | **1.97E+03** | 2.00E+03 | 4.00E+03 | 3.27E+03 | 8.39E+03 | 2.62E+03 | 9.91E+0[illegible] | 3.42E+05 | 4.13E+04 | 3.10E+04 |
| | *(1.21E+03)* | *(1.79E+03)* | *(2.82E+03)* | *(1.84E+03)* | *(3.82E+03)* | ***(1.18E+03)*** | *(8.83E+03)* | *(1.74E+05)* | *(2.39E+04)* | *(2.76E+04)* |
| *CEC-11* | **1.32E+03** | 1.38E+03 | 1.40E+03 | 1.35E+03 | 1.37E+03 | 1.39E+03 | 1.[illegible]E+03 | 1.41E+03 | 1.36E+03 | 1.65E+03 |
| | ***(1.34E+01)*** | *(2.42E+01)* | *(5.81E+01)* | *(1.12E+02)* | *(8.97E+01)* | *(5.42E[illegible])* | *([illegible]1E+02)* | *(7.73E+01)* | *(5.39E+01)* | *(3.00E+01)* |
| *CEC-12* | **1.30E+03** | 1.30E+03 | 1.30E+03 | 1.30E+03 | 1.30E+03 | 1.[illegible]+03 | 1.31E+03 | 1.31E+03 | 1.31E+03 | 1.30E+03 |
| | *(3.54E+00)* | *(7.89E-01)* | ***(6.69E-01)*** | *(6.94E-01)* | *(9.14E-01)* | *(8.07E-0[illegible])* | *(1.54E+00)* | *(2.05E+00)* | *(1.65E+00)* | *(9.36E-01)* |
| *CEC-13* | **1.30E+03** | 1.30E+03 | 1.30E+03 | 1.30E+03 | 1.30E+03 | [illegible] | 1.30E+03 | 1.35E+03 | 1.35E+03 | 1.30E+03 |
| | ***( 9.13E-07)*** | *( 2.76E-04)* | *(1.92E-04)* | *(5.44E-03)* | *(1.04E-03)* | *([illegible]3E-04)* | *(3.78E-03)* | *(4.70E+01)* | *(3.97E+01)* | *(4.96E-02)* |
| *CEC-14* | **3.78E+03** | 4.25E+03 | 7.29E+03 | 7.10E+03 | 7.60E+03 | 7.34E+[illegible] | 7.51E+03 | 9.30E+03 | 8.96E+03 | 6.08E+03 |
| | *(2.32E+03)* | *(1.73E+03)* | *(2.45E+03)* | *(3.12E+03)* | *(1.29E+0[illegible])* | [illegible]03) | *(1.52E+03)* | ***(1.94E+03)*** | *(6.32E+03)* | *(5.83E+03)* |
| *CEC-15* | **1.60E+03** | 1.60E+03 | 1.61E+03 | 1.60E+03 | 1.61E+03 | 1.60E+03 | 1.62E+03 | 1.64E+03 | 1.63E+03 | 1.61E+03 |
| | *(6.56E-02)* | *(3.76E+00)* | *(4.94E+00)* | ***(1.06E-02)*** | *(1.[illegible]01)* | *(1.80E-02)* | *(3.64E+00)* | *(1.12E+01)* | *(3.67E+01)* | *(1.08E+00)* |

at 0.05 significance level. First, the significant difference between two data sets has been observed by Mann-Whitney $U$ rank sum test. If there is no significant difference then sign '=' appears used when significant difference is observed then signs '+' or '-' appears for the case where SOA takes less or more function evaluations than the other competitor algorithms, respectively. In Table 10, '+' shows that SOA is significantly better and '-' shows that SOA is worse. As Table 10 includes 120 '+' signs out of 135 comparisons. Therefore, it can be concluded that the results of SOA is significantly effective than competitive approaches (i.e., SHO, GWO, PSO, MFO, MVO, SCA, GSA, GA, and DE) over CEC 2015 benchmark test suite. Overall, the results reveal that SOA outperforms other algorithms.

Table 9: $p-$values obtained from Wilcoxon rank sum test for CEC 2015 benchmark functions.

| *CEC* | SHO | GWO | PSO | MFO | MVO | SCA | GSA | GA | DE |
|---|---|---|---|---|---|---|---|---|---|
| *1* | 0.0045 | 0.0002 | 0.000[illegible] | 0.0002 | 0.0017 | 0.0003 | 0.0001 | 0.0005 | 0.0060 |
| *2* | 0.0010 | 0.0001 | 0.00[illegible] | 0.0060 | [illegible]0001 | 0.0075 | 0.0038 | 0.0021 | 0.0065 |
| *3* | 0.0001 | 0.0211 | 0.010[illegible] | 0.0179 | [illegible]0001 | 0.0043 | 0.0001 | 0.0101 | 0.0001 |
| *4* | 0.0065 | 0.0280 | 0.0264 | 0.00[illegible] | 0.0039 | 0.0045 | 0.0002 | 0.0097 | 0.0001 |
| *5* | 0.0006 | 0.0480 | [illegible]0001 | [illegible]007 | 0.0001 | 0.2413 | 0.0001 | 0.0009 | 0.0498 |
| *6* | 0.0001 | 0.0004 | 0.0[illegible]3 | 0.0050 | 0.8930 | 0.4726 | 0.0001 | 0.0046 | 0.0557 |
| *7* | 0.0092 | 0.0082 | 0.000[illegible] | 0.0033 | 0.0037 | 0.0028 | 0.0001 | 0.0072 | 0.0013 |
| *8* | 0.0093 | 0.0052 | 0.0001 | 0.0859 | 0.0031 | 0.0001 | 0.0233 | 0.0064 | 0.0087 |
| *9* | 0.0008 | 0.0011 | 0.0001 | 0.0211 | 0.0001 | 0.0001 | 0.0096 | 0.0192 | 0.0001 |
| *10* | 0.0001 | 0.0003 | 0.0001 | 0.0807 | 0.0001 | 0.0097 | 0.0001 | 0.0550 | 0.0043 |
| *11* | 0.0027 | 0.0113 | 0.0002 | 0.0001 | 0.0008 | 0.0490 | 0.0041 | 0.0005 | 0.0110 |
| *12* | 0.0001 | 0.0093 | 0.0001 | 0.0001 | 0.0008 | 0.0001 | 0.0001 | 0.0010 | 0.0259 |
| *13* | 0.0001 | 0.0048 | 0.0001 | 0.0001 | 0.0393 | 0.0001 | 0.0030 | 0.0022 | 0.0847 |
| *14* | 0.0001 | 0.0009 | 0.0045 | 0.0001 | 0.0896 | 0.0003 | 0.0001 | 0.0068 | 0.0036 |
| *15* | 0.0024 | 0.0012 | 0.0025 | 0.0497 | 0.0055 | 0.0001 | 0.0001 | 0.0032 | 0.0092 |

Table 10: Mann-Whitney $U$ rank sum test at 0.05 significance level (where '+' indicates SOA is significantly better, '-' indicates SOA is significantly worst and '=' indicates that there is no significant difference).

| F | SHO | GWO | PSO | MFO | MVO | SCA | GSA | GA | DE |
|---|---|---|---|---|---|---|---|---|---|
| *CEC-1* | + | + | + | + | + | + | + | + | + |
| *CEC-2* | - | + | - | - | - | + | - | - | - |
| *CEC-3* | + | + | + | + | + | + | = | + | = |
| *CEC-4* | + | + | - | + | + | + | + | + | + |
| *CEC-5* | - | - | - | + | + | + | + | + | + |
| *CEC-6* | + | + | - | + | + | + | + | + | + |
| *CEC-7* | + | + | + | + | + | + | + | + | + |
| *CEC-8* | + | + | + | + | + | + | + | + | + |
| *CEC-9* | + | + | + | + | - | + | + | + | + |
| *CEC-10* | + | + | + | + | + | + | + | + | + |
| *CEC-11* | + | + | + | + | + | + | + | + | + |
| *CEC-12* | + | + | + | + | + | + | + | + | + |
| *CEC-13* | + | + | + | + | + | + | + | + | + |
| *CEC-14* | + | + | + | + | + | + | + | + | + |
| *CEC-15* | + | + | + | + | + | + | + | + | + |

### *4.9. Investigating the influence of hyperparameters on SOA*

The hyperparameter tuning of metaheuristics is one of the major challenge. Different values of hyperparameter might yield different results as reported from the exiting literature. SOA algorithm involves three hyperparameters namely Maximum number of iterations, number of search agents, and parameter $f_c$.

The sensitivity investigation of these hyperparameters has been discussed by varying their values.

1. Maximum number of iterations: SOA algorithm was run for different number of iterations. The values of $Max_{iteration}$ used in experimentation are 100, 500, 800, and 1000. Fig. 10(a) reveals that SOA converges towards the optimum when the number of iterations is increased. The detailed analysis related to

iterations process is given in the Table 11.

2. Number of search agents: To investigate the effect of number of search agents on all test functions, SOA algorithm was executed for 50, 80, 100, and 200. For brevity, the $F11$ test function is chosen for analysis. Fig. 10(b) shows the effect of number of search agents. It was found that SOA provides best optimal solutions when the number of search agent is set to 100. The detailed results are given in Table 12.

3. Variation in parameter $f_c$: SOA algorithm was run for different values of parameter $f_c$ keeping other parameters fixed, i.e., maximum number of iterations and number of search agents. The values of $f_c$ used in experimentation are 1, 2, 3, 4, and 5. Fig. 10(c) shows that the function $F1$ obtains best optimal solution when the value of $f_c$ is set to 2. It is observed that $f_c = 2$ is a reasonable value for most of the test functions. The detailed sensitivity results related to $f_c$ are given in Table 13.

Table 11: The obtained optimal values on unimodal, multimod fixed-dim ension multimodal, CEC 2005, and CEC 2015 benchmark test functions ing different simulation runs. The best results are in bold.

| Functions | Iterations | Optimal val s |
|---|---|---|
| $F_1$ | 100 | 3.42E+00 |
| | 500 | 9.12E-1 |
| | 800 | 2.54E 0 |
| | **1000** | **0.00E+0** |
| $F_5$ | 100 | 3 1E+12 |
| | 500 | .37E+ 0 |
| | 800 | 08 |
| | **1000** | **5.18E** |
| $F_{11}$ | 100 | 8.14 +00 |
| | 500 | 6.14E 4 |
| | 800 | 5.70 04 |
| | | **0.00E+00** |
| $F_{17}$ | 100 | 1.19E+05 |
| | 500 | 1.52E+02 |
| | | 6.93E-02 |
| | **1000** | **1.14E-05** |
| $F_{23}$ | 00 | -2.01E+00 |
| | 500 | -9.81E+00 |
| | 00 | -1.02E+01 |
| | **1000** | **-1.10E+01** |
| $CF1$ | 100 | 1.80E+08 |
| | 500 | 3.13E+05 |
| | 800 | 4.91E+03 |
| | **1000** | **7.15E+02** |
| $CEC-1$ | 100 | 7.30E+12 |
| | 500 | 1.51E+09 |
| | 800 | 9.03E+05 |
| | **1000** | **2.22E+05** |

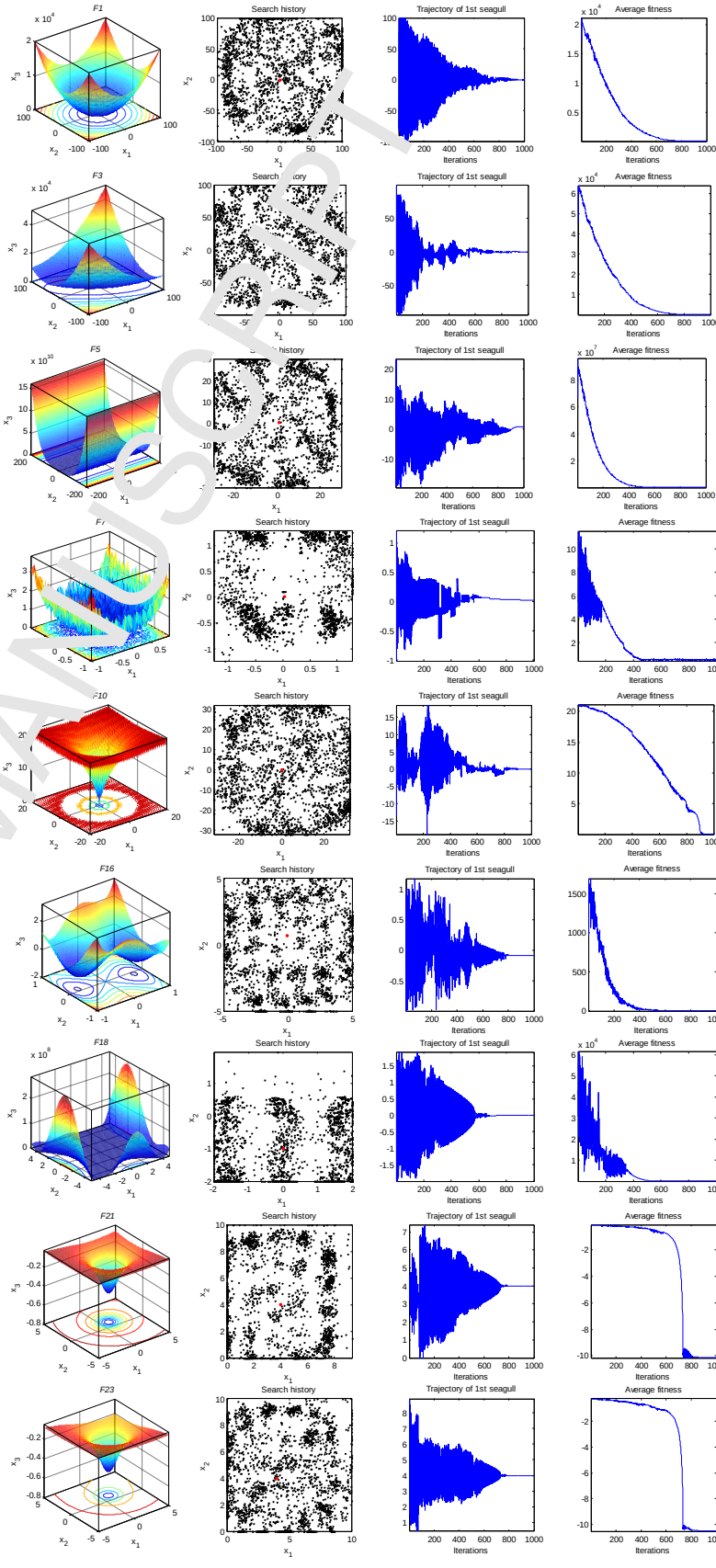



Figure 7: Search history, trajectory, and average fitness of SOA algorithm on 2D benchmark test problems.

Table 12: The obtained optimal values on unimodal, multimodal, fixed-dimension multimodal, CEC 2005, and CEC 2015 benchmark test functions where number of iterations is fixed as 1000. The number of search agents are varied from 50 to 200. The best results are in bold.

| Functions | Search agents | Optimal values |
|---|---|---|
| $F_1$ | **50** | **0.00E+00** |
| | **80** | **0.00E+00** |
| | **100** | **0.00E+00** |
| | **200** | **0.00E+00** |
| $F_5$ | 50 | 1.31E+01 |
| | 80 | 3.42E+02 |
| | **100** | **5.10E+00** |
| | 200 | 9.19E+00 |
| $F_{11}$ | **50** | **0.00E+00** |
| | **80** | **0.00E+00** |
| | **100** | **0.00E+00** |
| | **200** | **0.00E+00** |
| $F_{17}$ | 50 | 5.71E-03 |
| | 80 | 9.80E-03 |
| | **100** | **5.27E-04** |
| | 200 | 7.00E-02 |
| $F_{23}$ | 50 | -2.41E+00 |
| | 80 | -7.90E+0 |
| | **100** | **-2.72E+01** |
| | 200 | -4.11E+00 |
| $CF1$ | 50 | 6.1E+0 |
| | 80 | 8.7E+0 |
| | **100** | **2.72E+** |
| | 200 | 5.7 E+01 |
| $CEC-1$ | 50 | 1.23E+0 |
| | 80 | 1.50E+06 |
| | **100** | **1.34E+05** |
| | 200 | 8.57E+05 |

Table 13: The obtained optimal values on unimodal, multimodal, fixed-dimension multimodal, CEC 2005, and CEC 2015 benchmark test functions where number of iterations and search agents are fixed as 1000 and 100, respectively. The parameter $f_c$ is varied from 1 to 5. The best results are in bold.

| Functions | $f_c$ | Optimal values |
|---|---|---|
| $F_1$ | 1 | 2.45E-10 |
| | **2** | **0.00E+00** |
| | 3 | 7.74E-11 |
| | 4 | 4.30E-24 |
| | 5 | 4.00E-18 |
| $F_5$ | 1 | 4.40E+04 |
| | **2** | **8.00E+00** |
| | 3 | 2.01E+02 |
| | 4 | 3.98E+02 |
| | 5 | 3.00E+04 |
| $F_{11}$ | 1 | 4.00E-12 |
| | **2** | **0.00E+00** |
| | 3 | 1.73E-20 |
| | 4 | 4.38E-23 |
| | 5 | 3.60E-14 |
| $F_{17}$ | 1 | 1.21E+04 |
| | **2** | **8.90E-02** |
| | 3 | 4.11E+02 |
| | 4 | 9.78E+04 |
| | 5 | 2.09E+01 |
| $F_{23}$ | 1 | -2.41E+00 |
| | **2** | **-2.20E+02** |
| | 3 | -1.12E+01 |
| | 4 | -8.41E+00 |
| | 5 | -4.80E+01 |
| $CF1$ | 1 | 1.11E+03 |
| | **2** | **1.27E+01** |
| | 3 | 7.23E+02 |
| | 4 | 2.12E+03 |
| | 5 | 7.40E+03 |
| $CEC-1$ | 1 | 1.00E+07 |
| | **2** | **3.57E+03** |
| | 3 | 3.00E+07 |
| | 4 | 2.68E+10 |
| | 5 | 4.71E+08 |

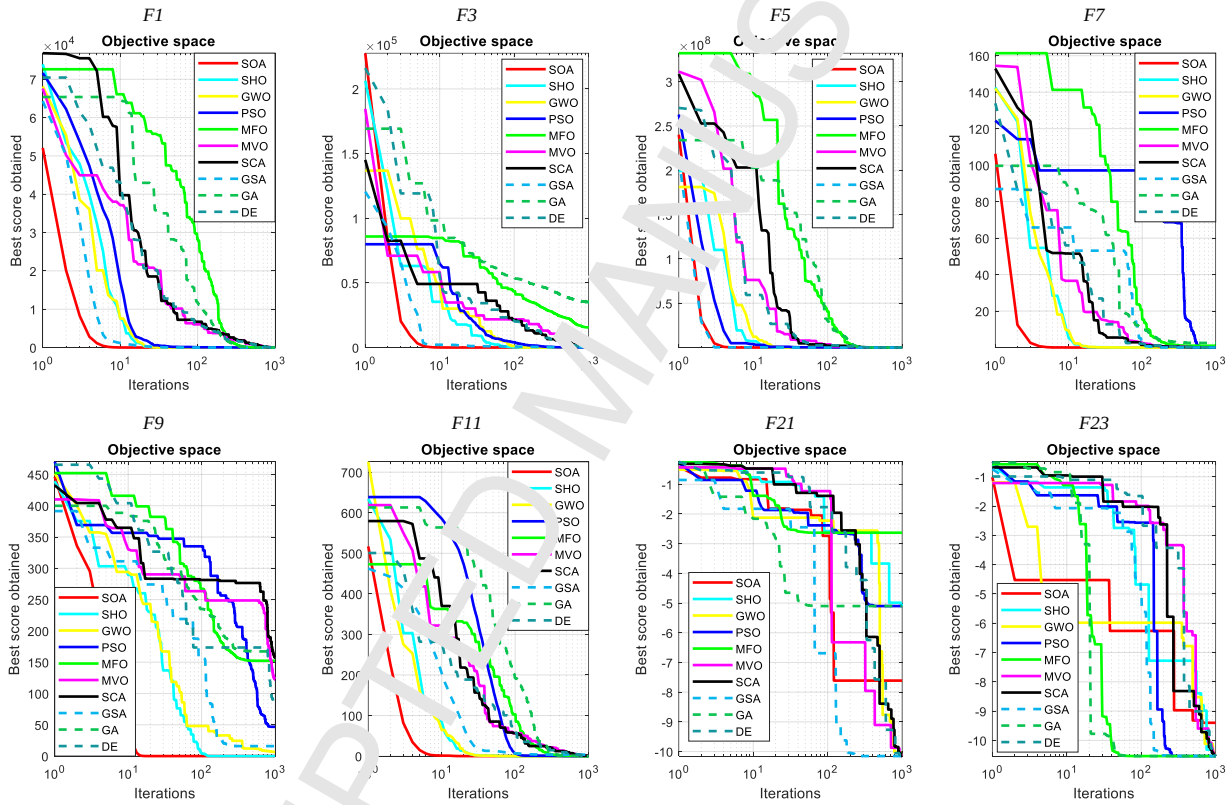



Figure 8: Convergence analysis of proposed SOA and competitor algorithms on some of the benchmark test functions.

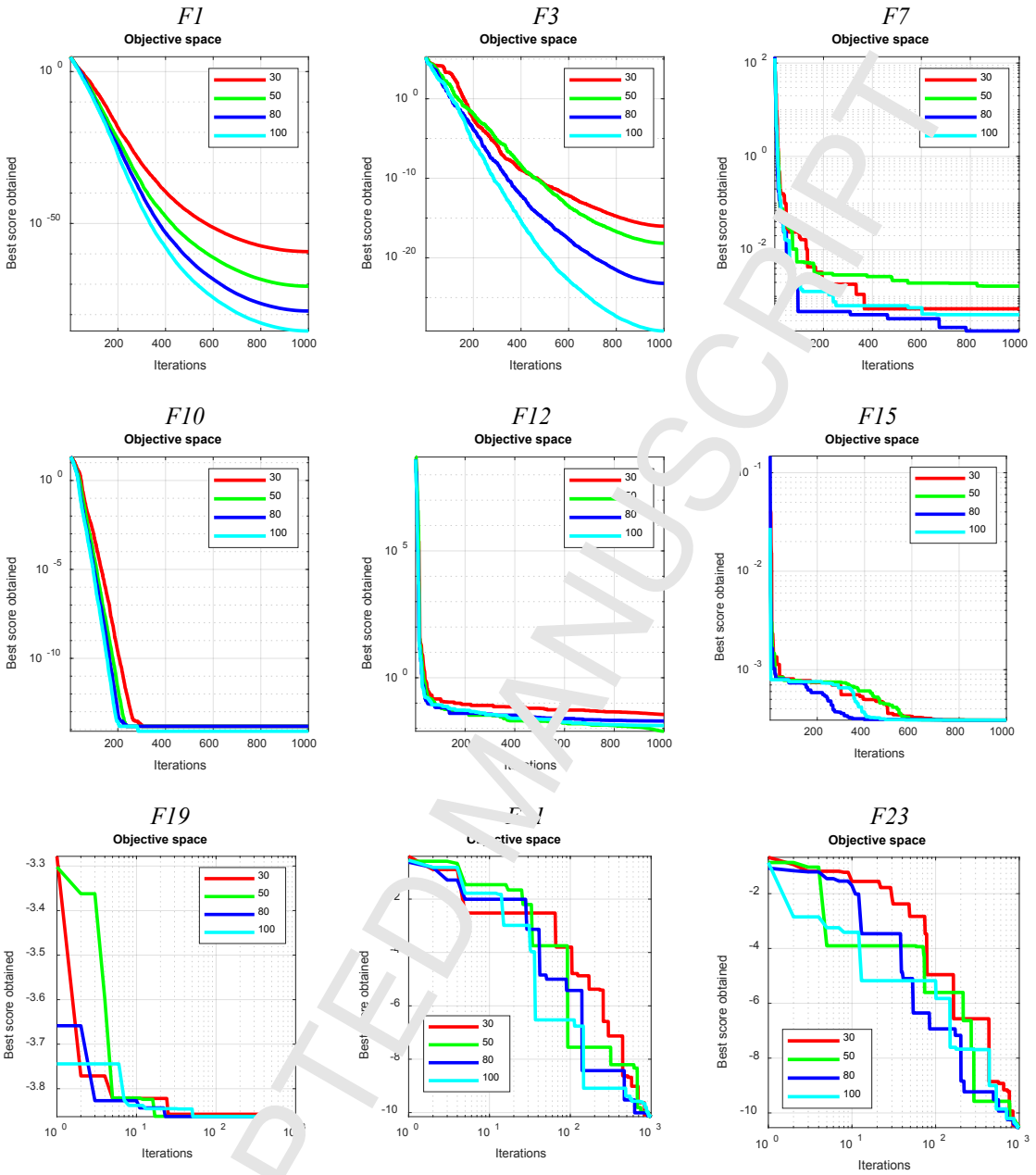



Figure 9: Sensitivity analysis of proposed SOA algorithm on some of the benchmark test functions.

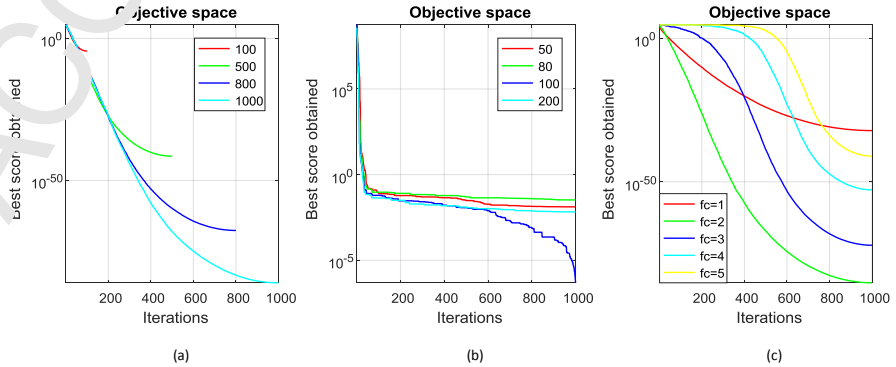



Figure 10: Hyperparameters analysis of proposed SOA algorithm, (a) Effect of iterations on $F1$ function; (b) Effect of search agents on $F11$ function; (c) Effect of parameter $f_c$ on $F1$ function.

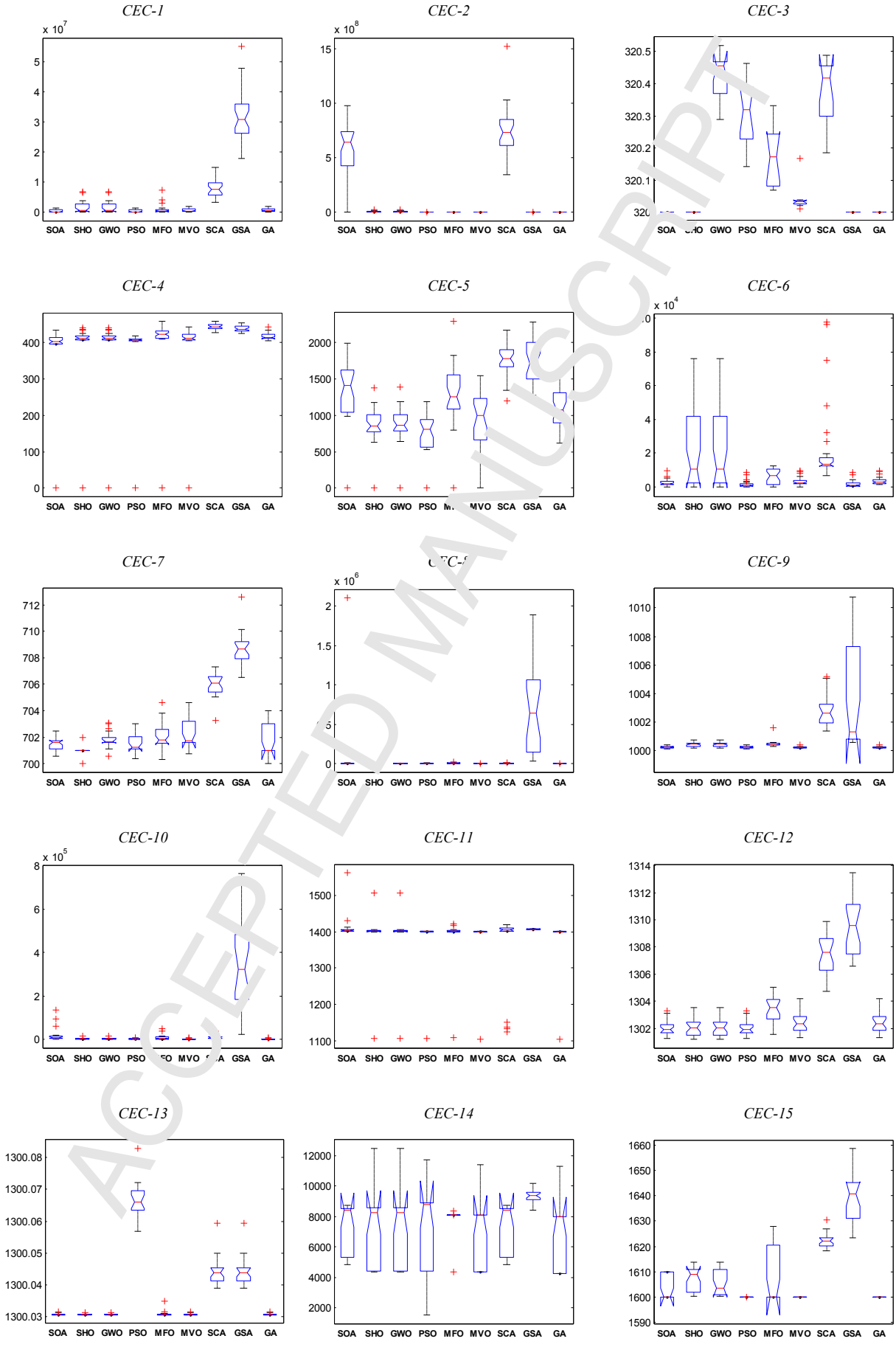



Figure 11: Box plot obtained from proposed SOA and other competitor algorithms on CEC 2015 benchmark test functions.

## 5. Real Industrial Engineering Applications

In this section, the efficiency of proposed SOA algorithm should be investigated in solving seven real-life constrained optimization problems and compared with other state-of-the-art algorithms. These problems have several equality and inequality constraints. SOA algorithm should be equipped with a constraint handling method to optimize these problems.

### 5.1. Constraint handling

Constraint handling is one of the biggest challenges in solving optimization problems using metaheuristic techniques. There are five constraint handling techniques [75]: penalty functions, hybrid methods, separation of objective functions and constraints, repair algorithms, and special operators. Among these techniques, the penalty functions are simple and easy to implement. There are numerous penalty functions such as static, annealing, adaptive, co-evolutionary, and death penalty. These approaches convert constraint problems into unconstraint problem by adding some penalty values. In this paper, a static penalty approach is employed to handle constraints in optimization problems.

$$\zeta(z) = f(z) \pm \Big[ \sum_{i=1}^{m} l_i \cdot max(0, t_i(z))^{\alpha} + \sum_{j=1}^{n} o_j \cdot \mid U_j(z) \mid^{\beta} \Big] \quad (15)$$

where $\zeta(z)$ is the modified objective function, $l_i$ and $o_j$ are positive penalty values, $t_i(z)$ and $U_j(z)$ are constraint functions, and $\alpha$ and $\beta$ are positive constants. The values of $\alpha$ and $\beta$ are 1 and 2, respectively. This approach assigns the penalty value for each infeasible solution. In death penalty approach, a large value is assigned to objective function of infeasible solution. Therefore, the static penalty function is employed which helps the search agents to move towards the feasible search space of the problem.

### 5.2. Optical buffer design problem

The optical buffer permits the optical CPUs to measure different optical packets during slow down the group velocity of light. This whole process is executed using most popular device known as Photonic Crystal Waveguide (PCW). Generally, PCWs have a lattice-shaped structure with a line defect and holes with different radii which yield the characteristics of slow light. In this subsection, the structure of PCW called a Bragg Slot Photonic Crystal Waveguide (BSPCW) is optimized to achieve these characteristics by SOA algorithm.

The performance of slow light devices is compared using Delay Bandwidth Product (DBP) and Normalized DBP (NDBP) metrics which are formulated as follows [76]:

$$DBP = \Delta d \cdot \Delta b \quad (16)$$

where $\Delta d$ and $\Delta b$ indicates the delay and bandwidth of slow light device, respectively.

$$NDBP = \overline{m_g} \cdot \Delta\omega/\omega_0 \quad (17)$$

where $\overline{m_g}$ is the average of group index, $\Delta\omega$ is the bandwidth, and $\omega_0$ is the central frequency of light wave. However, NDBP has a relation with group index ($m_g$) as:

$$m_g = \frac{V}{v_g} = C\frac{dv}{d\omega} \quad (18)$$

where $\omega$ is the dispersion, $v$ defines the wave vector, $C$ indicates the velocity of light, and $m_g$ is responsible for changing in the bandwidth range. The average of $m_g$ is calculated as follows:

$$\overline{m_g} = \int_{\omega_L}^{\omega_H} m_g(\omega)\frac{d\omega}{\Delta\omega} \quad (19)$$

Since, $m_g$ has a constant value with maximum fluctuation of $\pm 10\%$ [77]. The detailed information about PCWs can be found in [78]. The mathematical formulation of this problem is described as follows:

$$\begin{aligned}
&\vec{z} = [z_1 z_2 z_3 z_4 z_5 z_6 z_7 z_8] = [\frac{R_1}{a}\frac{R_2}{a}\frac{R_3}{a}\frac{R_4}{a}\frac{R_5}{a}\frac{l}{a}\frac{w_h}{a}\frac{w_l}{a}],\\
&\text{Maximize:} \quad f(\vec{z}) = NDBP,\\
&\text{Subject to:}\\
&max(|\beta_2(\omega)|) < 10^6 a/2\pi c^2,\\
&\omega_H < min(\omega_{band}),\\
&\omega_L > max(\omega_{lowband}),\\
&k_n \quad k_{nH} \text{ at } \omega_{guidedmode} > \omega_H,\\
&k_n < k_{nL} = \omega_{guidedmode} < \omega_L,\\
&\text{where,}\\
&v_g = \omega(k_{nH}) = \omega(1.1 m_{g0}), v\omega_L = \omega(k_{nL}) = \omega(0.9 m_{g0}),\\
&vk_n = \frac{ka}{2\pi}, \Delta\omega = \omega_H - \omega_L, a = \omega_0 \times 1550nm,\\
&0 \le z_{1-5} \le 0.5, \quad 0 \le z_6 \le 1, \quad 0 \le z_{7,8} \le 1.
\end{aligned} \quad (20)$$

There are five constraints in this problem for SOA algorithm. The algorithm is tested and run 30 times on this problem and the results obtained are tabulated in Table 14. The results reveal that there is a substantial 99% and 10% improvements achieved in bandwidth through proposed approach as compared with Wu *et al.* [79] and GWO [44], respectively. The similar behavior has been observed in NDBP. The improvements achieved in NDBP are 90% and 14% as compared with Wu *et al.* [79] and GWO [44] approaches, respectively. The optimized super cell is shown in Fig. 12. It shows that the optimized structure has a very good bandwidth without band mixing. The results demonstrate that SOA algorithm proved its merit for solving optical buffer optimization problem.

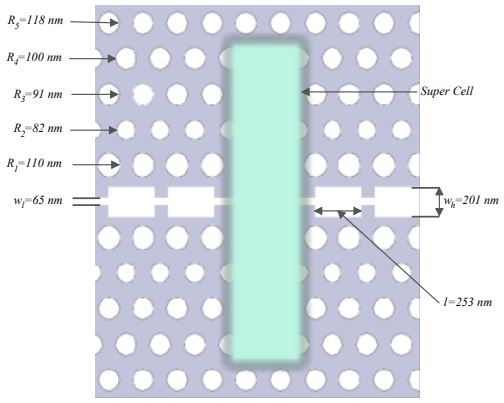



Figure 12: Optimized super cell using proposed SOA algorithm.

Table 14: Comparison results for optical buffer design problem.

| Parameters | SOA | GWO [44] | Wu *et al.* [79] |
|---|---|---|---|
| $R_1$ | 0.31348a | 0.33235a | - |
| $R_2$ | 0.22374a | 0.24952a | - |
| $R_3$ | 0.23400a | 0.26837a | - |
| $R_4$ | 0.27680a | 0.29498a | - |
| $R_5$ | 0.30217a | 0.34992a | - |
| $l$ | 0.7263a | 0.7437a | - |
| $w_h$ | 0.2052a | 0.2014a | - |
| $w_l$ | 0.60026a | 0.60073a | - |
| $a(nm)$ | 327 | 343 | 430 |
| $\overline{m_g}$ | 17.3 | 19.6 | 23 |
| *Bandwidth* | 37.1 | 33.9 | 17.6 |
| $\beta_2(a/2\pi c^2)$ | $10^3$ | $10^3$ | $10^3$ |
| *NDBP* | 0.49 | 0.43 | 0.26 |

### 5.3. Pressure vessel design problem

This problem was first proposed by Kannan and Kramer [80] to minimize the total cost consisting of material, forming, and welding of a cylindrical vessel. Fig. 13 shows the schematic view of pressure vessel which are capped at both ends by hemispherical heads. There are four variables in this problem:

- $T_s$ ($z_1$, thickness of the shell).
- $T_h$ ($z_2$, thickness of the head).
- $R$ ($z_3$, inner radius).
- $L$ ($z_4$, length of the cylindrical section without considering the head).

Among these four design variables, $R$ and $L$ are continuous variables. $T_s$ and $T_h$ are integer values which are multiples of 0.0625 *in*. The mathematical formulation of this problem is given below:

$$
\begin{aligned}
&\text{Consider } \vec{z} = [z_1\ z_2\ z_3\ z_4] = [T_s\ T_h\ R\ L],\\
&\text{Minimize } f(\vec{z}) = 0.6224 z_1 z_3 z_4 + 1.7781 z_2 z_3^2 + 3.1661 z_1^2 z_4 + 19.84 z_1^2 z_3,\\
&\text{Subject to:}\\
&g_1(\vec{z}) = -z_1 + 0.0193 z_3 \le 0,\\
&g_2(\vec{z}) = -z_3 + 0.00954 z_3 \le 0,\\
&g_3(\vec{z}) = -\pi z_3^2 z_4 - \frac{4}{3}\pi z_3^3 + 1{,}296{,}000 \le 0,\\
&g_4(\vec{z}) = z_4 - 240 \le 0,\\
&\text{where,}\\
&1 \times 0.0625 \le z_1,\ z_2 \le 99 \times 0.0625,\ 10.0 \le z_3,\ z_4 \le 200.0.
\end{aligned}
\tag{21}
$$

The comparison of obtained optimal solution from proposed SOA and other algorithms, i.e., SHO, GWO, PSO, MVO, SCA, GSA, GA, and DE is depicted in Table 15. The proposed SOA provides optimal solution at $z_{1-4} = (0.778080, 0.383247, 40.315120, 200.00000)$ with corresponding fitness value equal to $f(z_{1-4}) = 5879.5241$. It can be concluded that SOA is able to find best optimal design with minimum cost.

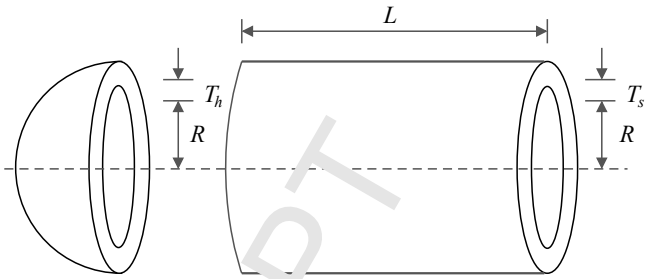



Figure 13: Schematic view of pressure vessel problem.

The statistical results of pressure vessel design problem are provided in Table 6. The results show that SOA performs better than the other competitor algorithms in terms of best, mean, and median. Fig. 14 shows the convergence behavior of optimal solution obtained from proposed SOA.

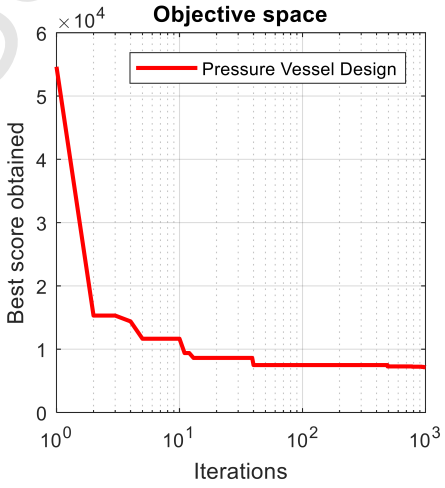



Figure 14: Convergence analysis of SOA for pressure vessel design problem.

### 5.4. Speed reducer design problem

The speed reducer design problem is a challenging benchmark due to its seven design variables [81] as shown in Fig. 15. The main objective of this problem is to minimize the weight of speed reducer with subject to constraints [82]:

- Bending stress of the gear teeth.
- Surface stress.
- Transverse deflections of the shafts.
- Stresses in the shafts.

There are seven design variables ($z_1 - z_7$) which can represent as the face width ($b$), module of teeth ($m$), number of teeth in the pinion ($p$), length of the first shaft between bearings ($l_1$), length of the second shaft between bearings ($l_2$), the diameter of first ($d_1$) shafts, and the diameter of second shafts ($d_2$). The third variable i.e., number of teeth in the pinion ($p$) is of integer values.

The mathematical formulation of this problem is formulated as follows:

$$
\begin{aligned}
&\text{Consider } \vec{z} = [z_1\ z_2\ z_3\ z_4\ z_5\ z_6\ z_7] = [b\ m\ p\ l_1\ l_2\ d_1\ d_2],\\
&\text{Minimize } f(\vec{z}) = 0.7854 z_1 z_2^2 (3.3333 z_3^2 + 14.9334 z_3 - 43.0934)\\
&\quad - 1.508 z_1 (z_6^2 + z_7^2) + 7.4777 (z_6^3 + z_7^3) + 0.7854 (z_4 z_6^2 + z_5 z_7^2),
\end{aligned}
\tag{22}
$$

Subject to:

$$g_1(\vec{z}) = \frac{27}{z_1 z_2^2 z_3} - 1 \le 0,$$

$$g_2(\vec{z}) = \frac{397.5}{z_1 z_2^2 z_3^2} - 1 \le 0,$$

$$g_3(\vec{z}) = \frac{1.93 z_4^3}{z_2 z_6^4 z_3} - 1 \le 0,$$

$$g_4(\vec{z}) = \frac{1.93 z_5^3}{z_2 z_7^4 z_3} - 1 \le 0,$$

$$g_5(\vec{z}) = \frac{[(745(z_4/z_2 z_3))^2 + 16.9 \times 10^6]^{1/2}}{110 z_6^3} - 1 \le 0,$$

$$g_6(\vec{z}) = \frac{[(745(z_5/z_2 z_3))^2 + 157.5 \times 10^6]^{1/2}}{85 z_7^3} - 1 \le 0,$$

$$g_7(\vec{z}) = \frac{z_2 z_3}{40} - 1 \le 0,$$

$$g_8(\vec{z}) = \frac{5 z_2}{z_1} - 1 \le 0,$$

$$g_9(\vec{z}) = \frac{z_1}{12 z_2} - 1 \le 0,$$

$$g_{10}(\vec{z}) = \frac{1.5 z_6 + 1.9}{z_4} - 1 \le 0,$$

$$g_{11}(\vec{z}) = \frac{1.1 z_7 + 1.9}{z_5} - 1 \le 0,$$

where,

$2.6 \le z_1 \le 3.6,\ 0.7 \le z_2 \le 0.8,\ 17 \le z_3 \le 28,\ 7.3 \le z_4 \le 8.3,$
$7.3 \le z_5 \le 8.3,\ 2.9 \le z_6 \le 3.9,\ 5.0 \le z_7 \le 5.5.$

The comparison of the obtained optimal solution with various competitor optimization algorithms is tabulated in Table 17. SOA provides optimal solution at $z_{1-7} = (3.50128, 0.7, 17, 7.3, 7.8, 3.33416, 5.24160)$ with corresponding fitness value equal to $f(z_{1-7}) = 2992.9985$. However, the statistical results of proposed and competitive optimization algorithms including SHO, GWO, PSO, MVO, SCA, GSA, WOA, and DE are given in Table 18.

The results indicate that SOA outperforms than other metaheuristics. Fig. 16 reveals that SOA algorithm converges towards the best optimal solution for speed reducer design problem.

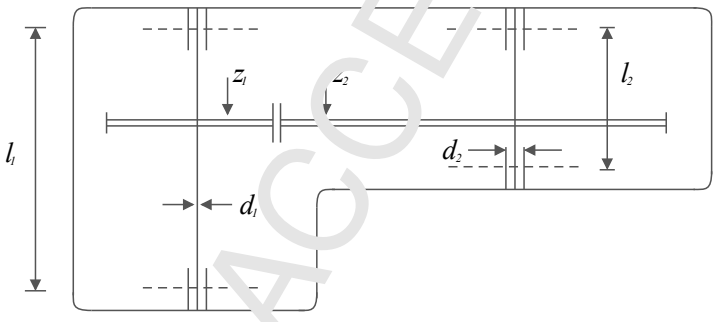



Figure 15: Schematic view of speed reducer problem.

### 5.5. Welded beam design problem

The main objective of this design problem is to minimize the fabrication cost of welded beam (see Fig. 17). The optimization constraints of welded beam are shear stress ($\tau$) and bending stress ($\theta$) in the beam, buckling load ($P_c$) on the bar, end deflection ($\delta$) of the beam. There are four design variables of this problem which are described as follows:

- $h$ ($z_1$, thickness of weld)
- $l$ ($z_2$, length of the clamped bar)
- $t$ ($z_3$, height of the bar)
- $b$ ($z_4$, thickness of the bar)

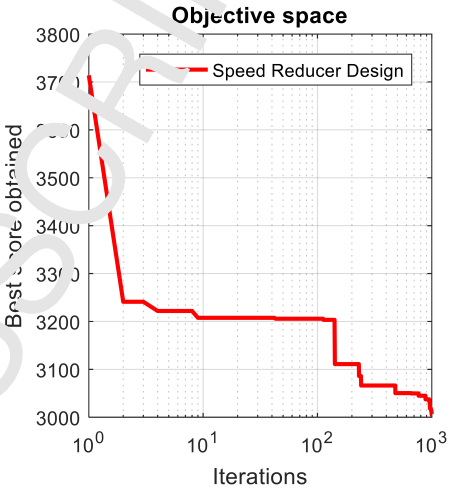



Figure 16: Convergence analysis of SOA for speed reducer design problem.

The mathematical formulation is described as follows:

Consider $\vec{z} = [z_1\ z_2\ z_3\ z_4] = [h\ l\ t\ b]$,
Minimize $f(\vec{z}) = 1.10471 z_1^2 z_2 + 0.04811 z_3 z_4 (14.0 + z_2)$,
Subject to:

$$g_1(\vec{z}) = \tau(\vec{z}) - 13,600 \le 0, \quad (23)$$
$$g_2(\vec{z}) = \sigma(\vec{z}) - 30,000 \le 0,$$
$$g_3(\vec{z}) = \delta(\vec{z}) - 0.25 \le 0,$$
$$g_4(\vec{z}) = z_1 - z_4 \le 0,$$

$$g_5(\vec{z}) = 6,000 - P_c(\vec{z}) \le 0,$$
$$g_6(\vec{z}) = 0.125 - z_1 \le 0,$$
$$g_7(\vec{z}) = 1.10471 z_1^2 + 0.04811 z_3 z_4 (14.0 + z_2) - 5.0 \le 0,$$

where,

$0.1 \le z_1,\ 0.1 \le z_2,\ z_3 \le 10.0,\ z_4 \le 2.0,$

$$\tau(\vec{z}) = \sqrt{(\tau')^2 + (\tau'')^2 + (l\tau'\tau'')/\sqrt{0.25(l^2 + (h+t)^2)}},$$

$$\tau' = \frac{6,000}{\sqrt{2} hl}, \qquad \sigma(\vec{z}) = \frac{504,000}{t^2 b}, \qquad \delta(\vec{z}) = \frac{65,856,000}{(30 \times 10^6) b t^3},$$

$$\tau'' = \frac{6,000(14 + 0.5l)\sqrt{0.25(l^2 + (h+t)^2)}}{2[0.707 hl(l^2/12 + 0.25(h+t)^2)]},$$

$$P_c(\vec{z}) = 64,746.022(1 - 0.0282346t) t b^3.$$

The comparison for best solution obtained from proposed SOA and other algorithms is depicted in Table 19. SOA provides optimal solution at $z_{1-4} = (0.205408, 3.472316, 9.035208, 0.201141)$ with corresponding fitness value equal to $f(z_{1-4}) = 1.723485$. The results indicate that SOA converges towards the best design and provides better solutions. The statistical comparison of the proposed SOA with other competitor algorithms is depicted in Table 20. It can be seen that SOA outperforms than other approaches in terms of best, mean, and median. Fig. 18 reveals the convergence analysis of best optimal solution obtained from proposed SOA algorithm.

Table 15: Comparison results for pressure vessel design problem.

| Algorithms | Optimum variables | | | | Optimum cost |
|---|---|---|---|---|---|
| | $T_s$ | $T_h$ | $R$ | $L$ | |
| SOA | 0.778080 | 0.383247 | 40.315120 | 200.00000 | **5879.5241** |
| SHO | 0.778210 | 0.384889 | 40.315040 | 200.0000[illegible] | 5885.5773 |
| GWO | 0.779035 | 0.384660 | 40.327793 | 199.6502[illegible] | 5889.3689 |
| PSO | 0.778961 | 0.384683 | 40.320913 | 200.00[illegible]0 | 5891.3879 |
| MVO | 0.845719 | 0.418564 | 43.816270 | 156.38164 | 6011.5148 |
| SCA | 0.817577 | 0.417932 | 41.74939 | 18[illegible].572[illegible]0 | 6137.3724 |
| GSA | 1.085800 | 0.949614 | 49.345231 | [illegible]9.48[illegible]41 | 11550.2976 |
| GA | 0.752362 | 0.399540 | 40.452514 | 198.0[illegible]268 | 5890.3279 |
| DE | 1.099523 | 0.906579 | 44.456397 | 179.6588[illegible] | 6550.0230 |

Table 16: Statistical results obtained from different algorithms for pressure vessel design problem.

| Algorithms | Best | Mean | Worst | Std. Dev. | Median |
|---|---|---|---|---|---|
| SOA | **5879.5241** | **5883.0052** | 5893.[illegible]21 | 256.415 | **5881.4486** |
| SHO | 5885.5773 | 5887.4441 | 5892.[illegible]3207 | **002.893** | 5886.2282 |
| GWO | 5889.3689 | 5891.5247 | 5894.623[illegible] | 013.910 | 5890.6497 |
| PSO | 5891.3879 | 6531.5032 | [illegible]4.3879 | 534.119 | 6416.1138 |
| MVO | 6011.5148 | 6477.3050 | 7250.[illegible]170 | 327.007 | 6397.4805 |
| SCA | 6137.3724 | 6326.7606 | [illegible]512.3541 | 126.609 | 6318.3179 |
| GSA | 11550.2976 | 23342.2909 | **33[illegible]26.2526** | 5790.625 | 24010.0415 |
| GA | 5890.3279 | 6264.0053 | 7[illegible]05.7500 | 496.128 | 6112.6899 |
| DE | 6550.0230 | 6643.9870 | [illegible]8005.4397 | 657.523 | 7586.0085 |

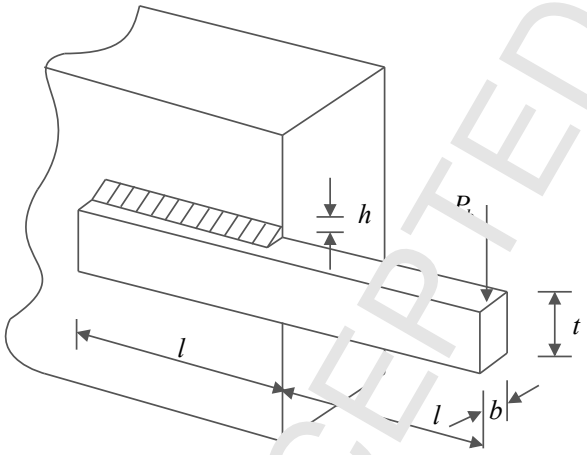



Figure 17: Schematic view of welded beam problem.

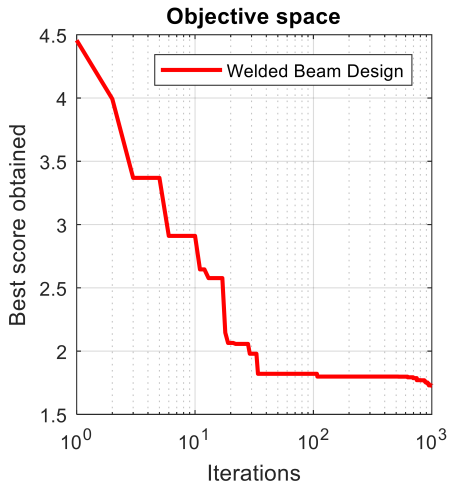



Figure 18: Convergence analysis of SOA for welded beam design problem.

### 5.6. Tension/compression spring design problem

The objective of this problem is to minimize the tension/ compression spring weight as shown in Fig. 19. The optimization constraints of this problem are described as follows:

- Shear stress.
- Surge frequency.
- Minimum deflection.

There are three design variables such as wire diameter ($d$), mean coil diameter ($D$), and the number of active coils ($P$).

The mathematical representation of this problem is described as follows:

$$\begin{aligned}
&\text{Consider } \vec{z} = [z_1\ z_2\ z_3] = [d\ D\ P],\\
&\text{Minimize } f(\vec{z}) = (z_3 + 2) z_2 z_1^2,\\
&\text{Subject to:}\\
&g_1(\vec{z}) = 1 - \frac{z_2^3 z_3}{71785 z_1^4} \leq 0,\\
&g_2(\vec{z}) = \frac{4z_2^2 - z_1 z_2}{12566(z_2 z_1^3 - z_1^4)} + \frac{1}{5108 z_1^2} \leq 0,
\end{aligned} \tag{24}$$

Table 17: Comparison results for speed reducer design problem.

| Algorithms | Optimum variables | | | | | | | Optimum cost |
|---|---|---|---|---|---|---|---|---|
| | $b$ | $m$ | $p$ | $l_1$ | $l_2$ | $d_1$ | $d_2$ | |
| SOA | 3.50128 | 0.7 | 17 | 7.3 | 7.8 | 3.33416 | 5.24160 | **2992.9985** |
| SHO | 3.50159 | 0.7 | 17 | 7.3 | 7.8 | 3.35127 | 5.2[illegible]874 | 2998.5507 |
| GWO | 3.506690 | 0.7 | 17 | 7.380933 | 7.815726 | 3.357847 | 5.286768 | 3001.288 |
| PSO | 3.500019 | 0.7 | 17 | 8.3 | 7.8 | 3.352412 | 5.2[illegible]6715 | 3005.763 |
| MVO | 3.508502 | 0.7 | 17 | 7.392843 | 7.816034 | 3.358073 | 5.286777 | 3002.928 |
| SCA | 3.508755 | 0.7 | 17 | 7.3 | 7.8 | 3.46[illegible]020 | 5.289213 | 3030.563 |
| GSA | 3.600000 | 0.7 | 17 | 8.3 | 7.8 | 3.3[illegible]9659 | 5.289224 | 3051.120 |
| GA | 3.510253 | 0.7 | 17 | 8.35 | 7.8 | 3.36222[illegible] | 5.287723 | 3067.561 |
| DE | 3.520124 | 0.7 | 17 | 8.37 | 7.8 | 3.366970 | 5.288719 | 3029.002 |

Table 18: Statistical results obtained from different algorithms for speed reducer design problem.

| Algorithms | Best | Mean | Worst | Std. Dev. | Median |
|---|---|---|---|---|---|
| SOA | **2992.9985** | **2996.756** | 2997.[illegible]25 | **1.78075** | **2991.997** |
| SHO | 2998.5507 | 2999.640 | 3003.889 | 1.93193 | 2999.187 |
| GWO | 3001.288 | 3005.845 | 3008.752 | 5.83794 | 3004.519 |
| PSO | 3005.763 | 3105.252 | 3[illegible]1.174 | 79.6381 | 3105.252 |
| MVO | 3002.928 | 3028.841 | 3060.[illegible]58 | 13.0186 | 3027.031 |
| SCA | 3030.563 | 3065.917 | 3104.779 | 18.0742 | 3065.609 |
| GSA | 3051.120 | 3170.334 | **3373.873** | 92.5726 | 3156.752 |
| GA | 3067.561 | 3186.523 | 3[illegible]13.199 | 17.1186 | 3198.187 |
| DE | 3029.002 | 3295.329 | 3619.465 | 57.0235 | 3288.657 |

$$g_3(\vec{z}) = 1 - \frac{140.45 z_1}{z_2^2 z_3} \leq 0,$$

$$g_4(\vec{z}) = \frac{z_1 + z_2}{1.5} - 1 \leq 0,$$

where,

$0.05 \leq z_1 \leq 2.0,\ 0.25 \leq z_2 \leq 1.3,\ 2.0 \leq z_3 \leq 15.$

The comparison between proposed SOA and other competitor algorithms is given in Table 21. SOA obtains best solution at design variables $z_{1-3} = (0.051065, 0.342897, 2.0[illegible]85)$ with an objective function value of $f(z_{1-3}) = 0.012643[illegible]17$. The results show that the proposed SOA again performs better than the other metaheuristic algorithms. The statistical results of tension/compression spring design are compared and tabulated in Table 22. It can be seen that SOA provides better statistical results than the other metaheuristics in terms of best, mean, and median.


Figure 19: Schematic view of tension/compression spring problem.

The convergence analysis of best optimal solution obtained from SOA is shown in Fig. 20.

### 5.7. 25-bar truss design problem

The truss design problem is a popular optimization problem as shown in Fig. 22. There are 10 nodes which are fixed and 25 bars

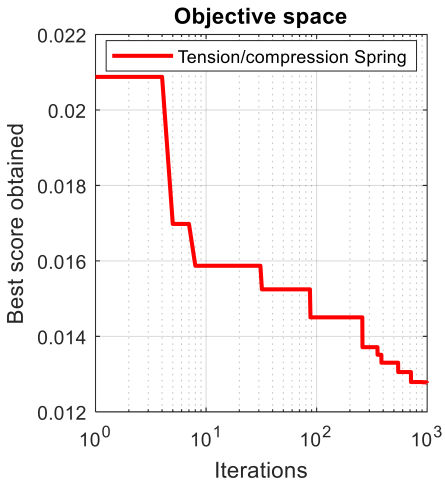



Figure 20: Convergence analysis of SOA for tension/compression spring design problem.

cross-sectional members which are grouped into eight categories.

- Group 1: $A_1$
- Group 2: $A_2, A_3, A_4, A_5$
- Group 3: $A_6, A_7, A_8, A_9$
- Group 4: $A_{10}, A_{11}$
- Group 5: $A_{12}, A_{13}$
- Group 6: $A_{14}, A_{15}, A_{17}$
- Group 7: $A_{18}, A_{19}, A_{20}, A_{21}$
- Group 8: $A_{22}, A_{23}, A_{24}, A_{25}$

The other variables which affects on this problem are as follows:

Table 19: Comparison results for welded beam design problem.

| Algorithms | Optimum variables | | | | Optimum cost |
|---|---|---|---|---|---|
| | $h$ | $l$ | $t$ | $b$ | |
| SOA | 0.205408 | 3.472316 | 9.035208 | 0.201141 | **1.723485** |
| SHO | 0.205563 | 3.474846 | 9.035799 | 0.205811 | 1.725661 |
| GWO | 0.205678 | 3.475403 | 9.036964 | 0.206229 | 1.726995 |
| PSO | 0.197411 | 3.315061 | 10.00000 | 0.201[illegible] | 1.820395 |
| MVO | 0.205611 | 3.472103 | 9.040931 | 0.205709 | 1.725472 |
| SCA | 0.204695 | 3.536291 | 9.004290 | 0.2[illegible] | 1.759173 |
| GSA | 0.147098 | 5.490744 | 10.00000 | 0.[illegible] | 2.172858 |
| GA | 0.164171 | 4.032541 | 10.00000 | 0.22[illegible] | 1.873971 |
| DE | 0.206487 | 3.635872 | 10.00000 | 0.203249 | 1.836250 |

Table 20: Statistical results obtained from different algorithms for welded beam design problem.

| Algorithms | Best | Mean | Worst | Std. Dev. | Median |
|---|---|---|---|---|---|
| SOA | **1.723485** | **1.724251** | 1.727[illegible] | 0.005967 | **1.724007** |
| SHO | 1.725661 | 1.725828 | 1.726064 | **0.000287** | 1.725787 |
| GWO | 1.726995 | 1.727128 | 1.727564 | 0.001157 | 1.727087 |
| PSO | 1.820395 | 2.230310 | **[illegible]** | 0.324525 | 2.244663 |
| MVO | 1.725472 | 1.729680 | 1.741[illegible] | 0.004866 | 1.727420 |
| SCA | 1.759173 | 1.817657 | 1.873408 | 0.027543 | 1.820128 |
| GSA | 2.172858 | 2.544239 | 3.0[illegible] | 0.255859 | 2.495114 |
| GA | 1.873971 | 2.119240 | 2.[illegible] | 0.034820 | 2.097048 |
| DE | 1.836250 | 1.363527 | 2.0[illegible] | 0.139485 | 1.9357485 |

Table 21: Comparison results for tension/compression spring design problem.

| Algorithms | Optimum variables | | | Optimum cost |
|---|---|---|---|---|
| | $d$ | $D$ | $P$ | |
| SOA | 0.051065 | 0.342897 | 12.0885 | **0.012645217** |
| SHO | 0.051144 | 0.343751 | 12.0955 | 0.012674000 |
| GWO | 0.050178 | 0.341541 | 12.07349 | 0.012678321 |
| PSO | 0.0500[illegible] | 0.310414 | 15.0000 | 0.013192580 |
| MVO | 0.05009 | 0.315956 | 14.22623 | 0.012816930 |
| SCA | 0.05[illegible]78[illegible] | 0.334779 | 12.72269 | 0.012709667 |
| GSA | 0.0[illegible]900 | 0.317312 | 14.22867 | 0.012873881 |
| GA | [illegible].05010 | 0.310111 | 14.0000 | 0.013036251 |
| DE | 0.05[illegible]025 | 0.316351 | 15.23960 | 0.012776352 |

Table 22: Statistical results obtained from different algorithms for tension/compression spring design problem.

| Algorithms | Best | Mean | Worst | Std. Dev. | Median |
|---|---|---|---|---|---|
| SOA | **0.012645217** | **0.012665871** | 0.012665417 | 0.001108 | **0.012655241** |
| SHO | [illegible].012674000 | 0.012684106 | 0.012715185 | **0.000027** | 0.012687293 |
| GWO | [illegible].012678321 | 0.012697116 | 0.012720757 | 0.000041 | 0.012699686 |
| PSO | 0.013192580 | 0.014817181 | **0.017862507** | 0.002272 | 0.013192580 |
| MVO | 0.012816930 | 0.014464372 | 0.017839737 | 0.001622 | 0.014021237 |
| SCA | 0.012709667 | 0.012839637 | 0.012998448 | 0.000078 | 0.012844664 |
| GSA | 0.012873881 | 0.013438871 | 0.014211731 | 0.000287 | 0.013367888 |
| GA | 0.013036251 | 0.014036254 | 0.016251423 | 0.002073 | 0.013002365 |
| DE | 0.012776352 | 0.013069872 | 0.015214230 | 0.000375 | 0.012952142 |

- $p$ = 0.0272 N/cm$^3$ (0.1 lb/in.$^3$)
- $E$ = 68947 MPa (10000 Ksi)
- Displacement limitation = 0.35 in.
- Maximum displacement = 0.3504 in.

- Design variable set = $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 2.2, 2.3, 2.4, 2.6, 2.8, 3.0, 3.2, 3.4\}$

Table 23 shows the member stress limitations for this truss problem. The loading conditions for 25-bar truss is presented in Table 24.

Table 23: Member stress limitations for 25-bar truss design problem.

| Element group | Compressive stress limitations Ksi (MPa) | Tensile stress limitations Ksi (MPa) |
|---|---|---|
| Group 1 | 35.092 (241.96) | 40.0 (275.80) |
| Group 2 | 11.590 (79.913) | 40.0 (275.80) |
| Group 3 | 17.305 (119.31) | 40.0 (275.80) |
| Group 4 | 35.092 (241.96) | 40.0 (275.80) |
| Group 5 | 35.092 (241.96) | 40.0 (275.80) |
| Group 6 | 6.759 (46.603) | 40.0 (275.80) |
| Group 7 | 6.959 (47.982) | 40.0 (275.80) |
| Group 8 | 11.082 (76.410) | 40.0 (275.80) |

The comparison of best solutions between several optimization algorithms is tabulated in Table 25. The proposed SOA is better than other competitor optimization algorithms in terms of best, average and standard deviation. SOA converges towards the best optimal design using low computational efforts (see Fig. 21).

### 5.8. Rolling element bearing design problem

The objective of this problem is to maximize the dynamic load carrying capacity of a rolling element bearing as demonstrated in Fig. 23. There are 10 decision variables such as pitch diameter ($D_m$), ball diameter ($D_b$), number of balls ($Z$), inner ($f_i$) and outer ($f_o$) raceway curvature coefficients, $K_{Dmin}$, $K_{Dmax}$, $e$, and $\zeta$. The mathematical representation of this problem is given below:

$$\text{Maximize } C_d = \begin{cases} f_c Z^{2/3} D_b^{1.8}, & \text{if } D < 25.4mm \\ C_d = 3.647 f_c Z^{2/3} D_b^{1.4} & \text{if } D > 25.4mm \end{cases}$$

Subject to:

$$g_1(\vec{z}) = \frac{\phi_0}{2sin^{-1}(D_b/D_m)} - Z + 1 \leq 0,$$
$$g_2(\vec{z}) = 2D_b - K_{Dmin}(D-d) \geq 0,$$
$$g_3(\vec{z}) = K_{Dmax}(D-d) - 2D_b \geq 0,$$
$$g_4(\vec{z}) = \zeta B_w - D_b \leq 0,$$
$$g_5(\vec{z}) = D_m - 0.5(D+d) \geq 0,$$
$$g_6(\vec{z}) = (0.5+e)(D+d) - D_m \geq 0,$$
$$g_7(\vec{z}) = 0.5(D - D_m - D_b) - \varepsilon D_b \geq 0,$$
$$g_8(\vec{z}) = f_i \geq 0.515,$$
$$g_9(\vec{z}) = f_o \geq 0.515, \quad (25)$$

where,

$$x = [\{(D-d)/2 - 3(T/4)\}^2 + \{D/2 - T/4 - D_b\}^2 - \{d/2 + T/4\}^2]$$
$$y = 2\{(D-d)/2 - 3(T/4)\}\{D/2 - T/4 - D_b\}$$
$$f_c = 37.91\left[1 + \left\{1.04\left(\frac{1-\gamma}{1+\gamma}\right)^{1.72}\left(\frac{f_i(2f_o-1)}{f_o(2f_i-1)}\right)^{0.41}\right\}^{10/3}\right]^{-0.3} \times \left[\frac{\gamma^{0.3}(1-\gamma)^{1.39}}{(1+\gamma)^{1/3}}\right]\left[\frac{2f_i}{2f_i-1}\right]^{0.41}$$

$$\phi_o = 2\pi - 2cos^{-1}\left(\frac{x}{y}\right)$$
$$\gamma = \frac{D_b}{D_m}, \quad f_i = \frac{r_i}{D_b}, \quad f_o = \frac{r_o}{D_b}, \quad T = D - d - 2D_b$$
$$D = 160, \quad d = 90, \quad \varepsilon = 0.5, \quad r_i = r_o = 11.033$$
$$0.5(D+d) \leq D_m \leq 0.6(D+d), \quad 0.15(D-d) \leq D_b \leq 0.45(D-d), \quad 4 \leq Z \leq 50, \quad 0.515 \leq f_i \text{ and } f_o \leq 0.6,$$
$$0.4 \leq K_{Dmin} \leq 0.5, \quad 0.6 \leq K_{Dmax} \leq 0.7, \quad 0.3 \leq e \leq 0.4,$$
$$0.02 \leq e \leq 0.1, \quad 0.6 \leq \zeta \leq 0.85.$$

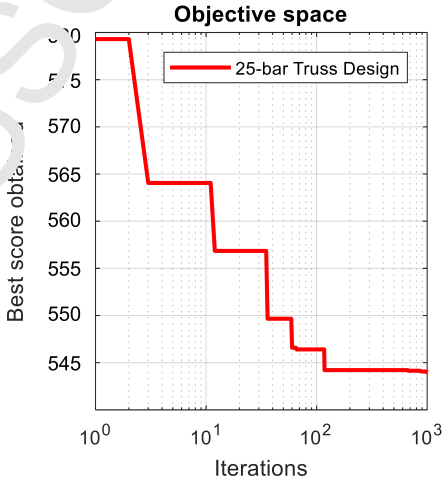



Figure 21: Convergence analysis of SOA for 25-bar truss design problem.

Table 26 reveals the performance comparison between the proposed SOA and other competitor algorithms. SOA provides optimal solution at $z_{1-10} = (125, 21.41892, 10.94123, 0.515, 0.515, 0.4, 0.7, 0.3, 0.02, 0.6)$ with corresponding fitness value equal to $f(z_{1-10}) = 85068.052$. The statistical results obtained for rolling element bearing design problem are compared and given in Table 27. Fig. 24 shows the convergence analysis of SOA algorithm. The results reveals that SOA is capable to achieve a best optimal solution.

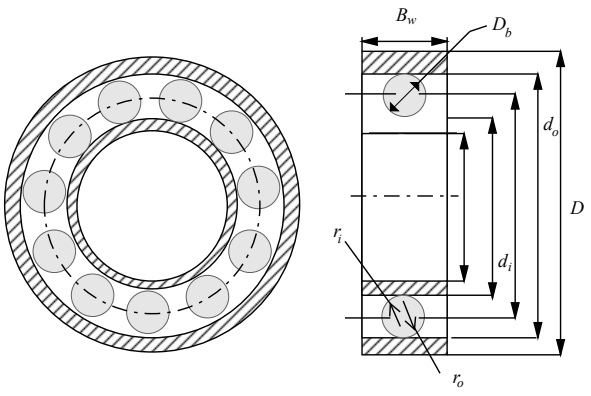



Figure 23: Schematic view of rolling element bearing problem.

Table 24: Two loading conditions for the 25-bar truss design problem.

| Node | Case 1 | | | Case 2 | | |
|---|---|---|---|---|---|---|
| | $P_xKips(kN)$ | $P_yKips(kN)$ | $P_zKips(kN)$ | $P_xKips(kN)$ | $P_yKips(kN)$ | $P_zKips(kN)$ |
| 1 | 0.0 | 20.0 (89) | -5.0 (22.25) | 1.0 (4.45) | 10.0 (44.5) | -5.0 (22.25) |
| 2 | 0.0 | -20.0 (89) | -5.0 (22.25) | 0.0 | -20.0 (44.5) | -5.0 (22.25) |
| 3 | 0.0 | 0.0 | 0.0 | 0.5 (2.22) | 0.0 | 0.0 |
| 6 | 0.0 | 0.0 | 0.0 | 0.5 (2.22) | 0.0 | 0.0 |

Table 25: Statistical results obtained from different algorithms for 25-bar truss design problem.

| Groups | SOA | ACO [83] | PSO [84] | GSS [85] | BB-BC [86] |
|---|---|---|---|---|---|
| $A1$ | **0.01** | 0.01 | 0.01 | 0.01 | 0.01 |
| $A2-A5$ | **1.897** | 2.042 | 2.052 | 2.003 | 1.993 |
| $A6-A9$ | **3.002** | 3.001 | 3.001 | 3.007 | 3.056 |
| $A10-A11$ | **0.01** | 0.01 | 0.01 | 0.01 | 0.01 |
| $A12-A13$ | **0.01** | 0.01 | 0.01 | 0.01 | 0.01 |
| $A14-A17$ | **0.663** | 0.684 | 0.684 | 0.687 | 0.665 |
| $A18-A21$ | 1.621 | 1.625 | **1.621** | 1.655 | 1.642 |
| $A22-A25$ | 2.671 | 2.672 | 2.673 | **2.66** | 2.679 |
| **Best weight** | **544.85** | 545.03 | 545.21 | 545.10 | 545.16 |
| **Average weight** | **545.08** | 545.74 | 546.84 | 545.58 | 545.66 |
| **Std. dev.** | **0.391** | 0.94 | 1.48 | 0.412 | 0.491 |

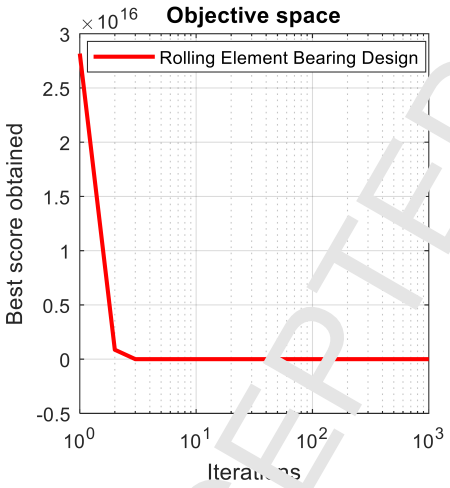



Figure 24: Convergence analysis of SOA for rolling element bearing design problem.

In summary, these results show that SOA is an effective optimizer for solving large scale constrained engineering design problems with low computational cost and fast convergence speed.

## 6. Conclusion and future works

This paper presents a novel bio-inspired based optimization algorithm called Seagull Optimization Algorithm (SOA). In this work, SOA is tested on forty-four benchmark test functions to validate its efficiency. The results reveal that SOA provides very competitive results as compared with nine well-known optimization algorithms. The results on the unimodal and multimodal test functions show the exploration and exploitation capabilities of SOA algorithm, respectively. Whereas, results of the CEC 2005 and CEC 2015 benchmark test functions show that SOA is able to solve challenging and high dimensionality bound constrained real problems. The statistical testing has been performed to demonstrate the statistical significance of proposed SOA over benchmark test functions. In addition, SOA is employed to seven real-life industrial applications (i.e., Optical buffer design, Pressure vessel design, Speed reducer design, Welded beam design, Tension/compression spring design, 25-bar truss design, and Rolling element bearing design problems) to verify and demonstrate its performance.

For future works, parallel seagull optimization algorithm can be developed to automatically detects the number of available processors and divides the workload among them to accomplish the effective utilization of the available resources. OpenMP library is used to speed up the performance and efficiency of traditional seagull optimization algorithm. The binary and multi-objective versions of SOA algorithm can also be developed to solve large scale real-life engineering problems.

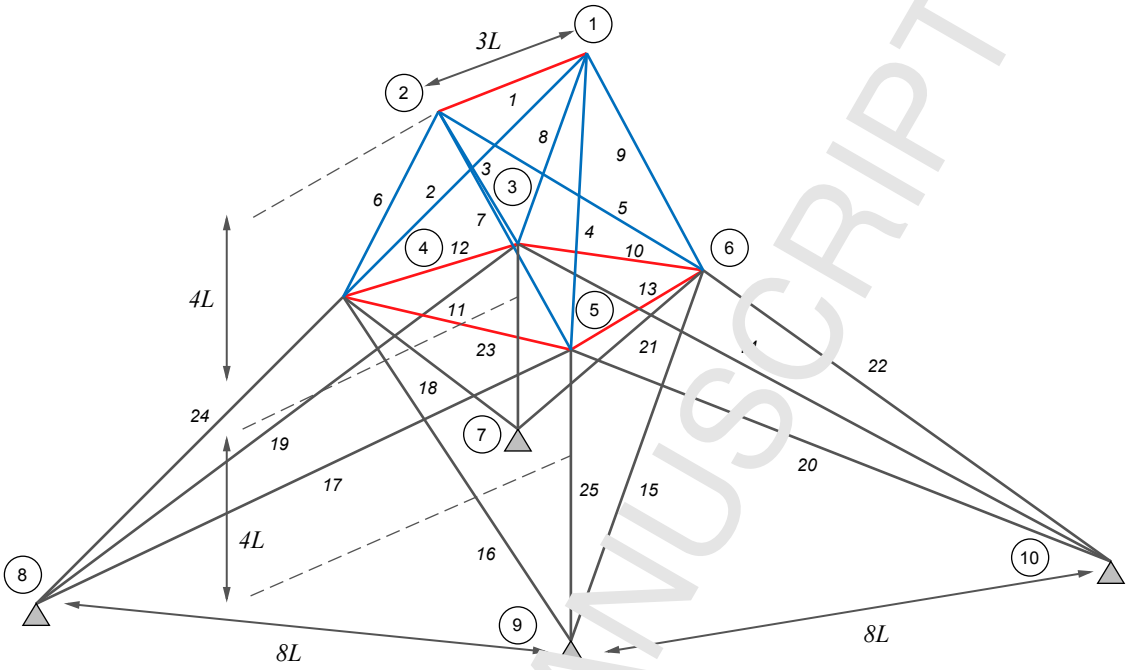



Figure 22: Schematic view of 25-bar truss problem.

Table 26: Comparison results for rolling element bearing design problem.

| Algorithms | Optimum variables | | | | | | | | | | Opt. cost |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | $D_m$ | $D_b$ | $Z$ | $f_i$ | $f_o$ | $K_{Dmin}$ | $K_{Dmax}$ | $\varepsilon$ | $e$ | $\zeta$ | |
| SOA | 125 | 21.41892 | 10.94123 | 0.515 | 0.515 | 0.4 | 0.7 | 0.3 | 0.02 | 0.6 | **85068.052** |
| SHO | 125 | 21.40732 | 10.93268 | 0.515 | 0.515 | 0.4 | 0.7 | 0.3 | 0.02 | 0.6 | 85054.532 |
| GWO | 125.6199 | 21.35129 | 10.98781 | 0.515 | 0.515 | 0.5 | 0.68807 | 0.300151 | 0.03254 | 0.62701 | 84807.111 |
| PSO | 125 | 20.75388 | 11.173 2 | 0.515 | 0.515000 | 0.5 | 0.61503 | 0.300000 | 0.05161 | 0.60000 | 81691.202 |
| MVO | 125.6002 | 21.32250 | 10.97 3 | 0.515 | 0.515000 | 0.5 | 0.68782 | 0.301348 | 0.03617 | 0.61061 | 84491.266 |
| SCA | 125 | 21.14834 | 10.96928 | 0.515 | 0.515 | 0.5 | 0.7 | 0.3 | 0.02778 | 0.62912 | 83431.117 |
| GSA | 125 | 20.85417 | 11.15 89 | 0.515 | 0.517746 | 0.5 | 0.61827 | 0.304068 | 0.02000 | 0.624638 | 82276.941 |
| GA | 125 | 20.77562 | 1.011 47 | 0.515 | 0.515000 | 0.5 | 0.61397 | 0.300000 | 0.05004 | 0.610001 | 82773.982 |
| DE | 125 | 20.87123 | 11.1 697 | 0.515 | 0.516000 | 0.5 | 0.61951 | 0.301128 | 0.05024 | 0.614531 | 81569.527 |

## Appendix A Unimodal, Multimodal, and Fixed-dimension multimodal benchmark test functions.

### *A.1 Unimodal benchmark test functions*

#### *A.1.1 Sphere Model*

$$F_1(z) = \sum_{i=1}^{} z_i^2$$

$-100 \le z_i \le 100, \quad f_{min} = 0, \quad Dim = 30$

#### *A.1.2 Schwefel's Problem 2.22*

$$F_2(z) = \sum_{i=1}^{30} |z_i| + \prod_{i=1}^{30} |z_i|$$

$-10 \le z_i \le 10, \quad f_{min} = 0, \quad Dim = 30$

#### *A.1.3 Schwefel's Problem 1.2*

$$F_3(z) = \sum_{i=1}^{30} \left( \sum_{j=1}^{i} z_j \right)^2$$

$-100 \le z_i \le 100, \quad f_{min} = 0, \quad Dim = 30$

#### *A.1.4 Schwefel's Problem 2.21*

$$F_4(z) = max_i\{|z_i|, 1 \le i \le 30\}$$

$-100 \le z_i \le 100, \quad f_{min} = 0, \quad Dim = 30$

#### *A.1.5 Generalized Rosenbrock's Function*

Table 27: Statistical results obtained from different algorithms for rolling element bearing design problem.

| Algorithms | Best | Mean | Worst | Std. Dev. | Median |
|---|---|---|---|---|---|
| SOA | **85068.052** | **85044.954** | 86540.853 | 1523.76 | **85055.421** |
| SHO | 85054.532 | 85024.858 | 85853.876 | 0186.68 | 85040.241 |
| GWO | 84807.111 | 84791.613 | 84517.923 | 0137.186 | 84960.147 |
| PSO | 81691.202 | 50435.017 | **32761.546** | **13962.15** | 42287.581 |
| MVO | 84491.266 | 84353.685 | 84100.834 | 0392.4 | 84398.601 |
| SCA | 83431.117 | 81005.232 | 77992.482 | 1710.777 | 81035.109 |
| GSA | 82276.941 | 78002.107 | 71043.110 | 31 9.90 | 78398.853 |
| GA | 82773.982 | 81198.753 | 80687.239 | 79.3 7 | 8439.728 |
| DE | 81569.527 | 80397.998 | 79412.779 | 1756. 2 | 8347.009 |

$$F_5(z) = \sum_{i=1}^{29}[100(z_{i+1} - z_i^2)^2 + (z_i - 1)^2]$$

$-30 \le z_i \le 30, \quad f_{min} = 0, \quad Dim = 30$

### A.1.6 Step Function

$$F_6(z) = \sum_{i=1}^{30}(\lfloor z_i + 0.5 \rfloor)^2$$

$-100 \le z_i \le 100, \quad f_{min} = 0, \quad Dim = 30$

### A.1.7 Quartic Function

$$F_7(z) = \sum_{i=1}^{30} i z_i^4 + random[0, 1]$$

$-1.28 \le z_i \le 1.28, \quad f_{min} = 0, \quad Dim = 30$

## A.2 Multimodal benchmark test functions

### A.2.1 Generalized Schwefel's Problem 2.2

$$F_8(z) = \sum_{i=1}^{30} - z_i sin(\sqrt{|z_i|})$$

$-500 \le z_i \le 500, \quad f_{min} = -12569.5, \quad Dim = 30$

### A.2.2 Generalized Rastrigin's Function

$$F_9(z) = \sum_{i=1}^{30}[z_i^2 - 10cos(2\pi z_i) + 10]$$

$-5.12 \le z_i \le 5.12, \quad f_{min} = 0, \quad Dim = 30$

### A.2.3 Ackley's Function

$$F_{10}(z) = -20exp\Big(-0.2\sqrt{\frac{1}{30}\sum_{i=1}^{30} z_i^2}\Big) - exp\Big(\frac{1}{30}\sum_{i=1}^{30} cos(2\pi z_i)\Big) + 20 + e$$

$-32 \le z_i \le 32, \quad f_{min} = 0, \quad Dim = 30$

### A.2.4 Generalized Griewank Function

$$F_{11}(z) = \frac{1}{4000}\sum_{i=1}^{30} z_i^2 - \prod_{i=1}^{30} cos\Big(\frac{z_i}{\sqrt{i}}\Big) + 1$$

$-600 \le z_i \le 600, \quad f_{min} = 0, \quad Dim = 30$

### A.2.5 Generalized Penalized Functions

- $F_{12}(z) = \frac{\pi}{30}\{10sin(\pi x_1) + \sum_{i=1}^{29}(x_i - 1)^2[1 + 10sin^2(\pi x_{i+1})] + (x_n - 1)^2\} + \sum_{i=1}^{30} u(z_i, 10, 100, 4)$

  $-50 \le z_i \le 50, \quad f_{min} = 0, \quad Dim = 30$

- $F_{13}(z) = 0.1\{sin^2(3\pi z_1) + \sum_{i=1}^{29}(z_i - 1)^2[1 + sin^2(3\pi z_i + 1)] + (z_n - 1)^2[1 + sin^2(2\pi z_{30})]\} + \sum_{i=1}^{N} u(z_i, 5, 100, 4)$

  $-50 \le z_i \le 50, \quad f_{min} = 0, \quad Dim = 30$

  where, $x_i = 1 + \frac{z_i + 1}{4}$

$$u(z_i, a, k, m) = \begin{cases} k(z_i - a)^m & z_i > a \\ 0 & -a < z_i < a \\ k(-z_i - a)^m & z_i < -a \end{cases}$$

## A.3 Fixed-dimension multimodal benchmark test functions

### A.3.1 Shekel's Foxholes Function

$$F_{14}(z) = \Big(\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^{2}(z_i - a_{ij})^6}\Big)^{-1}$$

$-65.536 \le z_i \le 65.536, \quad f_{min} \approx 1, \quad Dim = 2$

### A.3.2 Kowalik's Function

$$F_{15}(z) = \sum_{i=1}^{11}\Big[a_i - \frac{z_1(b_i^2 + b_i z_2)}{b_i^2 + b_i z_3 + z_4}\Big]^2$$

$-5 \le z_i \le 5, \quad f_{min} \approx 0.0003075, \quad Dim = 4$

Table 28: Shekel's Foxholes Function $F_{14}$.

| $(a_{ij}, i = 1, 2$ and $j = 1, 2, \ldots, 25)$ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $i \backslash j$ | 1 | 2 | 3 | 4 | 5 | 6 | ... | 25 |
| 1 | -32 | -16 | 0 | 16 | 32 | -32 | ... | 32 |
| 2 | -32 | -32 | -32 | -32 | -32 | -16 | ... | 32 |

### *A.3.3 Six-Hump Camel-Back Function*

$$F_{16}(z) = 4z_1^2 - 2.1z_1^4 + \frac{1}{3}z_1^6 + z_1z_2 - 4z_2^2 + 4z_2^4$$

$$-5 \leq z_i \leq 5, \quad f_{min} = -1.0316285, \quad Dim = 2$$

### *A.3.4 Branin Function*

$$F_{17}(z) = \left(z_2 - \frac{5.1}{4\pi^2}z_1^2 + \frac{5}{\pi}z_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)cosz_1 + 10$$

$$-5 \leq z_1 \leq 10, \quad 0 \leq z_2 \leq 15, \quad f_{min} = 0.398, \quad Dim = 2$$

### *A.3.5 Goldstein-Price Function*

$F_{18}(z) = [1 + (z_1 + z_2 + 1)^2(19 - 14z_1 + 3z_1^2 - 14z_2 + 6z_1z_2 + 3z_2^2)] \times [30 + (2z_1 - 3z_2)^2 \times (18 - 32z_1 + 12z_1^2 + 48z_2 - 36z_1z_2 + 27z_2^2)]$

$$-2 \leq z_i \leq 2, \quad f_{min} = 3, \quad Dim = 2$$

### *A.3.6 Hartman's Family*

- $F_{19}(z) = -\sum_{i=1}^{4} c_i exp(-\sum_{j=1}^{3} a_{ij}(z_j - p_{ij})^2)$

  $0 \leq z_j \leq 1, \quad f_{min} = -3.86, \quad Dim = 3$

- $F_{20}(z) = -\sum_{i=1}^{4} c_i exp(-\sum_{j=1}^{6} a_{ij}(z_j - p_{ij})^2)$

  $0 \leq z_j \leq 1, \quad f_{min} = -3.32, \quad Dim = 6$

Table 29: Hartman Function $F_{19}$.

| $i$ | $(a_{ij}, j = 1, 2, 3)$ | | | $c_i$ | $(p_{ij}, j = 1, 2, 3)$ | | |
|---|---|---|---|---|---|---|---|
| 1 | 3 | 10 | 30 | 1 | 0.3689 | 0.1170 | 0.2673 |
| 2 | 0.1 | 10 | 35 | 1.2 | 0.4699 | 0.4387 | 0.7470 |
| 3 | 3 | 10 | 30 | 3 | 0.1091 | 0.8732 | 0.5547 |
| 4 | 0.1 | 10 | 35 | 3.2 | 0.038150 | 0.5743 | 0.8828 |

### *A.3.7 Shekel's Foxholes Function*

- $F_{21}(z) = -\sum_{i=1}^{5}[(X - a_i)(X - a_i)^T + c_i]^{-1}$

  $0 \leq z_i \leq 10, \quad f_{min} = -10.1532, \quad Dim = 4$

- $F_{22}(z) = -\sum_{i=1}^{7}[(X - a_i)(X - a_i)^T + c_i]^{-1}$

  $0 \leq z_i \leq 10, \quad f_{min} = -10.4028, \quad Dim = 4$

- $F_{23}(z) = -\sum_{i=1}^{10}[(X - a_i)(X - a_i)^T + c_i]^{-1}$

  $0 \leq z_i \leq 10, \quad f_{min} = -10.536, \quad Dim = 4$

Table 30: Shekel Foxholes Functions $F_{21}, F_{22}, F_{23}$.

| $i$ | $(a_{ij}, j = 1, 2, 3, 4)$ | | | | $c_i$ |
|---|---|---|---|---|---|
| 1 | 4 | 4 | 4 | 4 | 0.1 |
| 2 | 1 | 1 | 1 | 1 | 0.2 |
| 3 | 8 | 8 | 8 | 8 | 0.2 |
| 4 | 6 | 6 | 6 | 6 | 0.4 |
| 5 | 3 | 7 | 3 | 7 | 0.4 |
| 6 | 2 | 9 | 2 | 9 | 0.6 |
| 7 | 5 | 5 | 3 | 3 | 0.3 |
| 8 | 8 | 1 | 8 | 1 | 0.7 |
| 9 | 6 | 2 | 6 | 2 | 0.5 |
| 10 | 7 | 3.6 | 7 | 3.6 | 0.5 |

## *A.4 Basic composite benchmark test functions*

### *A.4.1 Weierstrass Function*

$$F(z) = \sum_{i=1}^{30}\Big(\sum_{k=0}^{20}[0.5^k cos(2\pi 3^k(z_i + 0.5))]\Big) - 30\sum_{k=0}^{20}[0.5^k cos(2\pi 3^k \times 0.5)]$$

Note that the Sphere, Rastrigin's, Griewank's, and Ackley's functions in composite benchmark suite are same as above mentioned $F_1, F_9, F_{11}$, and $F_{10}$ benchmark test functions.

## *A.5 Basic CEC 2015 benchmark test functions*

### *A.5.1 Bent Cigar Function*

$$F(z) = z_1^2 + 10^6\sum_{i=2}^{30} z_i^2$$

### *A.5.2 Discus Function*

$$F(z) = 10^6 z_1^2 + \sum_{i=2}^{30} z_i^2$$

### *A.5.3 Modified Schwefel's Function*

$$F(z) = 418.9829 \times 30 - \sum_{i=1}^{30} g(y_i), \quad y_i = z_i + 4.209687462275036e + 002$$

Table 31: Hartman Function $F_{20}$.

| $i$ | $(a_{ij}, j = 1, 2, \ldots, 6)$ | | | | | | $c_i$ | $(p_{ij}, j = 1, 2, \ldots, 6)$ | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 10 | 3 | 17 | 3.5 | 1.7 | 8 | 1 | 0.1312 | 0.1696 | 0.5569 | 0.0124 | 0.82[illegible]3 | [illegible]5886 |
| 2 | 0.05 | 10 | 17 | 0.1 | 8 | 14 | 1.2 | 0.2329 | 0.4135 | 0.8307 | 0.3736 | 0.1004 | 0.9991 |
| 3 | 3 | 3.5 | 1.7 | 10 | 17 | 8 | 3 | 0.2348 | 0.1415 | 0.3522 | 0.2883 | [illegible].3047 | 0.6650 |
| 4 | 17 | 8 | 0.05 | 10 | 0.1 | 14 | 3.2 | 0.4047 | 0.8828 | 0.8732 | 0.5743 | [illegible]0[illegible]1 | 0.0381 |

$$g(y_i) = \begin{cases} y_i sin(|y_i|^{1/2}) \\ \quad where,\ if\ |y_i| \le 500, \\ (500 - mod(y_i, 500))sin(\sqrt{|500 - mod(y_i, 500)|}) - \dfrac{(y_i - 500)^2}{10000 \times 30} \\ \quad where,\ if\ y_i > 500, \\ (mod(|y_i|, 500) - 500)sin(\sqrt{|mod(|y_i|, 500) - 500|}) - \dfrac{(y_i + 500)^2}{10000 \times 30} \\ \quad where,\ if\ y_i < -500 \end{cases}$$

### A.5.4 Katsuura Function

$$F(z) = \frac{10}{30^2}\prod_{i=1}^{30}\Big(1 + i\sum_{j=1}^{32}\frac{|2^j z_i - round(2^j z_i)|}{2^j}\Big)^{\frac{10}{30^{1.2}}} - \frac{10}{30^2}$$

### A.5.5 HappyCat Function

$$F(z) = \Big|\sum_{i=1}^{30} z_i^2 - 30\Big|^{1/4} + (0.5\sum_{i=1}^{30} z_i^2 + \sum_{i=1}^{30} z_i)/30 + 0.5$$

### A.5.6 HGBat Function

$$F(z) = \Big|(\sum_{i=1}^{30} z_i^2)^2 - (\sum_{i=1}^{30} z_i)^2\Big|^{1/2} + (0.5\sum_{i=1}^{30} z_i^2 + \sum_{i=1}^{30} z_i)/30 + 0.5$$

### A.5.7 Expanded Griewank's plus Rosenbrock's Function

$$F(z) = F_{39}(F_{38}(z_1, z_2)) + F_{39}(F_{38}(z_2, z_3)) + \ldots + F_{39}(F_{38}(z_{30}, z_1))$$

### A.5.8 Expanded Scaffer's F6 Function

Scaffer's F6 Function:

$$g(z, y) = 0.5 + \frac{(sin^2(\sqrt{z^2 + x^2}) - 0.5)}{(1 + 0.001(z^2 + x^2))^2}$$

$$F(z) = g(z_1, z_2) + g(z_2, z_3) + \ldots + g(z_{30}, z_1)$$

### A.5.9 High Conditioned Elliptic Function

$$F(z) = \sum_{i=1}^{30}(10^6)^{\frac{i-1}{30-1}} z_i^2$$

Note that the Weierstrass, Rosenbrock's, Griewank's, Rastrigin's, and Ackley's functions in CEC 2015 benchmark test suite are same as above mentioned $Weierstrass$, $F_5$, $F_{11}$, $F_9$, and $F_{10}$ benchmark test functions.

### A.4. Composite benchmark functions

The detailed description of six well-known composite benchmark test functions ($F_{24} - F_{29}$) are mentioned in Table 32.

Table 32: Composite benchmark test functions.

| Functions | Dim | Range | $f_{min}$ |
|---|---|---|---|
| $F_{24}(CF1)$ :<br>$f_1, f_2, f_3, \ldots, f_{10}$ = Sphere Function<br>$[\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_{10}] = [1, 1, 1, \ldots, 1]$<br>$[\beta_1, \beta_2, \beta_3, \ldots, \beta_{10}] = [5/100, 5/100, 5/100, \ldots, 5/100]$ | 10 | [-5, 5] | 0 |
| $F_{25}(CF2)$ :<br>$f_1, f_2, f_3, \ldots, f_{10}$ = Griewank's Function<br>$[\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_{10}] = [1, 1, 1, \ldots, 1]$<br>$[\beta_1, \beta_2, \beta_3, \ldots, \beta_{10}] = [5/100, 5/100, 5/100, \ldots, 5/100]$ | 10 | [-5, 5] | 0 |
| $F_{26}(CF3)$ :<br>$f_1, f_2, f_3, \ldots, f_{10}$ = Griewank's Function<br>$[\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_{10}] = [1, 1, 1, \ldots, 1]$<br>$[\beta_1, \beta_2, \beta_3, \ldots, \beta_{10}] = [1, 1, 1, \ldots, 1]$ | 10 | [-5, 5] | 0 |
| $F_{27}(CF4)$ :<br>$f_1, f_2$ = Ackley's Function<br>$f_3, f_4$ = Rastrigin's Function<br>$f_5, f_6$ = Weierstras's Function<br>$f_7, f_8$ = Griewank's Function<br>$f_9, f_{10}$ = Sphere Function<br>$[\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_{10}] = [1, 1, 1, \ldots, 1]$<br>$[\beta_1, \beta_2, \beta_3, \ldots, \beta_{10}] = [5/32, 5/32, 1, 1, 5/0.5, 5/0.5, 5/100, 5/100, 5/100, 5/100]$ | 10 | [-5, 5] | 0 |
| $F_{28}(CF5)$ :<br>$f_1, f_2$ = Rastrigin's Function<br>$f_3, f_4$ = Weierstras's Function<br>$f_5, f_6$ = Griewank's Function<br>$f_7, f_8$ = Ackley's Function<br>$f_9, f_{10}$ = Sphere Function<br>$[\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_{10}] = [1, 1, 1, \ldots, 1]$<br>$[\beta_1, \beta_2, \beta_3, \ldots, \beta_{10}] = [1/5, 1/5, 5/0.5, 5/0.5, 5/100, 5/100, 5/32, 5/32, 5/100, 5/100]$ | 10 | [-5, 5] | 0 |
| $F_{29}(CF6)$ :<br>$f_1, f_2$ = Rastrigin's Function<br>$f_3, f_4$ = Weierstras's Function<br>$f_5, f_6$ = Griewank's Function<br>$f_7, f_8$ = Ackley's Function<br>$f_9, f_{10}$ = Sphere Function<br>$[\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_{10}] = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]$<br>$[\beta_1, \beta_2, \beta_3, \ldots, \beta_{10}] =$<br>$[0.1 * 1/5, 0.2 * 1/5, 0.3 * 5/0.5, 0.4 * 5/0.5, 0.5 * 5/100, 0.6 * 5/100, 0.7 * 5/32, 0.8 * 5/32, 0.9 * 5/100, 1 * 5/100]$ | 10 | [-5, 5] | 0 |

### A.5. CEC 2015 benchmark test functions

The detailed description of fifteen well-known CEC 2015 benchmark test functions ($CEC1 - CEC15$) are mentioned in Table 33.

Table 33: CEC 2015 benchmark test functions.

| No. | Functions | Related basic functions | Dim | $f_{min}$ |
|---|---|---|---|---|
| $CEC-1$ | Rotated Bent Cigar Function | Bent Cigar Function | 30 | 100 |
| $CEC-2$ | Rotated Discus Function | Discus Function | 30 | 200 |
| $CEC-3$ | Shifted and Rotated Weierstrass Function | Weierstrass Function | 30 | 300 |
| $CEC-4$ | Shifted and Rotated Schwefel's Function | Schwefel's Function | 30 | 400 |
| $CEC-5$ | Shifted and Rotated Katsuura Function | Katsuura Function | 30 | 500 |
| $CEC-6$ | Shifted and Rotated HappyCat Function | HappyCat Function | 30 | 600 |
| $CEC-7$ | Shifted and Rotated HGBat Function | HGBat Function | 30 | 700 |
| $CEC-8$ | Shifted and Rotated Expanded Griewank's plus Rosenbrock's Function | Griewank's Function<br>Rosenbrock's Function | 30 | 800 |
| $CEC-9$ | Shifted and Rotated Expanded Scaffer's F6 Function | Expanded Scaffer's F6 Function | 30 | 900 |
| $CEC-10$ | Hybrid Function 1 ($N = 3$) | Schwefel's Function<br>Rastrigin's Function<br>High Conditioned Elliptic Function | 30 | 1000 |
| $CEC-11$ | Hybrid Function 2 ($N = 4$) | Griewank's Function<br>Weierstrass Function<br>Rosenbrock's Function<br>Scaffer's F6 Function | 30 | 1100 |
| $CEC-12$ | Hybrid Function 3 ($N = 5$) | Katsuura Function<br>HappyCat Function<br>Expanded Griewank's plus Rosenbrock's Function<br>Schwefel's Function<br>Ackley's Function | 30 | 1200 |
| $CEC-13$ | Composition Function 1 ($N = 5$) | Rosenbrock's Function<br>High Conditioned Elliptic Function<br>Bent Cigar Function<br>Discus Function<br>High Conditioned Elliptic Function | 30 | 1300 |
| $CEC-14$ | Composition Function 2 ($N = 3$) | Schwefel's Function<br>Rastrigin's Function<br>High Conditioned Elliptic Function | 30 | 1400 |
| $CEC-15$ | Composition Function 3 ($N = 5$) | HGBat Function<br>Rastrigin's Function<br>Schwefel's Function<br>Weierstrass Function<br>High Conditioned Elliptic Function | 30 | 1500 |

## References


[1] Zheng Yan, Jun Wang, and Guocheng Li. A collective neurodynamic optimization approach to bound-constrained nonconvex optimization. *Neural networks*, 55:20–29, 2014.

[2] Rajesh Kumar Chandrawat, Rakesh Kumar, BP Garg, Gaurav Dhiman, and Sumit Kumar. An analysis of modeling and optimization production cost through fuzzy linear programming problem with symmetric and right angle triangular fuzzy number. In *Proceedings of Sixth International Conference on Soft Computing for Problem Solving*, pages 197–211. Springer, 2017.

[3] Pritpal Singh and Gaurav Dhiman. A fuzzy-LP approach in time series forecasting. In *International Conference on Pattern Recognition and Machine Intelligence*, pages 243–253. Springer, 2017.

[4] Amandeep Kaur and Gaurav Dhiman. A review on search-based tools and techniques to identify bad code smells in object-oriented systems. In *Harmony Search and Nature Inspired Optimization Algorithms*, pages 909–921. Springer, 2019.

[5] Pritpal Singh and Gaurav Dhiman. Uncertainty representation using fuzzy-entropy approach: Special application in remotely sensed high-resolution satellite images (RSHRS). *Applied Soft Computing*, 72:121 – 139, 2018.

[6] Pritpal Singh, Kinjal Rabadiya, and Gaurav Dhiman. A four way decision-making system for the indian summer monsoon rainfall. *Modern Physics Letters B*, page 1850004, 2018.

[7] Gaurav Dhiman and Vijay Kumar. Spotted hyena optimizer for solving complex and non-linear constrained engineering problems. In *Harmony Search and Nature Inspired Optimization Algorithms*, pages 857–867. Springer, 2019.

[8] Gaurav Dhiman and Amandeep Kaur. A hybrid algorithm based on particle swarm and spotted hyena optimizer for global optimization. In *Advances in Intelligent Systems and Computing*. Springer, 2018, In press.

[9] Gaurav Dhiman and Amandeep Kaur. Spotted hyena optimizer for solving engineering design problems. In *Machine Learning and Data Science (MLDS), 2017 International Conference on*, pages 114–119. IEEE, 2017.

[10] Gaurav Dhiman and Vijay Kumar. Multi-objective spotted hyena optimizer: A multi-objective optimization algorithm for engineering problems. *Knowledge-Based Systems*, 150:175 – 197, 2018.

[11] Pritpal Singh and Gaurav Dhiman. A hybrid fuzzy time series forecasting model based on granular computing and bio-inspired optimization approaches. *Journal of Computational Science*, 27:370 – 385, 2018.

[12] Amandeep Kaur, Satnam Kaur, and Gaurav Dhiman. A quantum method for dynamic nonlinear programming technique using schrödinger equation and monte carlo approach. *Modern Physics Letters B*, page 1850374, 2018.

[13] Gaurav Dhiman and Amandeep Kaur. Optimizing the design of airfoil and optical buffer problems using spotted hyena optimizer. *Designs*, 2(3):28, 2018.

[14] Gaurav Dhiman and Vijay Kumar. KnRVEA: A hybrid evolutionary algorithm based on knee points and reference vector adaptation strategies for many-objective optimization. *Applied Intelligence, In press*, 2018.

[15] James Kennedy and Russell C. Eberhart. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, pages 1942–1948, 1995.

[16] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. Ant colony optimization - artificial ants as a computational intelligence technique. *IEEE Computational Intelligence Magazine*, 1:28–39, 2006.

[17] E. Alba and B. Dorronsoro. The exploration/exploitation tradeoff in dynamic cellular genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 9(2):126–142, 2005.

[18] M. Lozano and C. Garcia-Martinez. Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: Overview and progress report. *Computers and Operations Research*, 37(3):481–497, 2010.

[19] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[20] Pritpal Singh, Gaurav Dhiman, and Amandeep Kaur. A quantum approach for time series data based on graph and schrodinger equations methods. *Modern Physics Letters A, In press*, 2018.

[21] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[22] Esmat Rashedi, Hossein Nezamabadi-pour, and Saeid Saryazdi. GSA: A gravitational search algorithm. *Information Sciences*, 179(13):2232 – 2248, 2009.

[23] Osman K. Erol and Ibrahim Eksin. A new optimization method: Big bang-big crunch. *Advances in Engineering Software*, 37(2):106 – 111, 2006.

[24] A. Kaveh and S. Talatahari. A novel heuristic optimization method: charged system search. *Acta Mechanica*, 213(3):267–289, 2010.

[25] Abdolreza Hatamlou. Black hole: A new heuristic optimization approach for data clustering. *Information Sciences*, 222:175 – 184, 2013.

[26] Richard A. Formato. Central force optimization: A new deterministic gradient-like optimization metaheuristic. *Opsearch*, 46(1):25–51, 2009.

[27] Haifeng Du, Xiaodong Wu, and Jian Zhuang. Small-world optimization algorithm for function optimization. *Springer Berlin Heidelberg*, pages 264–273, 2006.

[28] A. Kaveh and M. Khayatazad. A new meta-heuristic method: Ray optimization. *Computers and Structures*, 112-113:283 – 294, 2012.

[29] John H Holland. Genetic algorithms. *Scientific american*, 267(1):66–72, 1992.

[30] Rainer Storn and Kenneth Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.

[31] John R. Koza. Genetic programming: On the programming of computers by means of natural selection. *MIT Press*, 1992.

[32] Hans-Georg Beyer and Hans-Paul Schwefel. Evolution strategies – a comprehensive introduction. *Natural Computing*, 1(1):3–52, 2002.

[33] D. Simon. Biogeography-based optimization. *IEEE Transactions on Evolutionary Computation*, 12(6):702–713, 2008.

[34] Gaurav Dhiman and Vijay Kumar. Emperor penguin optimizer: A bio-inspired algorithm for engineering problems. *Knowledge-Based Systems*, 159:20–50, 2018.

[35] Jianchao Fan and Jun Wang. A collective neurodynamic optimization approach to nonnegative matrix factorization. *IEEE transactions on neural networks and learning system*, 28(10):2344–2356, 2017.

[36] Zheng Yan, Jianchao Fan, and Jun Wang. A collective neurodynamic approach to constrained global optimization. *IEEE transactions on neural networks and learning systems*, 28(5):1206–1215, 2017.

[37] Tiancai Wang, Xing He, Tingwen Huang, Chuandong Li, and Wei Zhang. Collective neurodynamic optimization for economic emission dispatch problem considering valve point effect in microgrid. *Neural Networks*, 93:126–136, 2017.

[38] Bilal Alatas. Acroa: Artificial chemical reaction optimization algorithm for global optimization. *Expert Systems with Applications*, 38(10):13170 – 13180, 2011.

[39] Hamed Shah Hosseini. Principal components analysis by the galaxy-based search algorithm: a novel metaheuristic for continuous optimisation. *International Journal of Computational Science and Engineering*, 6:132–140, 2011.

[40] Xin-She Yang. Firefly algorithm, stochastic test functions and design optimisation. *International Journal of Bio-Inspired Computation*, 2(2):78–84, 2010.

[41] Naser Ghorbani and Ebrahim Babaei. Exchange market algorithm. *Applied Soft Computing*, 19:177 – 187, 2014.

[42] Fatemeh Ramezani and Shahriar Lotfi. Social-based algorithm. *Applied Soft Computing*, 13(5):2837 – 2856, 2013.

[43] Zong W. Geem, Joong H. Kim, and G. V. Loganathan. A new heuristic optimization algorithm: Harmony search. *Simulation*, 76(2):60–68, February 2001.

[44] Seyedali Mirjalili, Seyed Mohammad Mirjalili, and Andrew Lewis. Grey wolf optimizer. *Advances in Engineering Software*, 69:46 – 61, 2014.

[45] Ali Sadollah, Ardeshir Bahreininejad, Hadi Eskandar, and Mohd Hamdi. Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems. *Applied Soft Computing*, 13(5):2592 – 2612, 2013.

[46] Antonio Mucherino and Onur Seref. Monkey search: a novel metaheuristic search for global optimization. *AIP Conference Proceedings*, 953(1), 2007.

[47] Swagatam Das, Arijit Biswas, Sambarta Dasgupta, and Ajith Abraham. *Bacterial Foraging Optimization Algorithm: Theoretical Foundations, Analysis, and Applications*, pages 23–55. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[48] Wen-Tsao Pan. A new fruit fly optimization algorithm: Taking the financial distress model as an example. *Knowledge-Based Systems*, 26:69 – 74, 2012.

[49] Alireza Askarzadeh. Bird mating optimizer: An optimization algorithm inspired by bird mating strategies. *Communications in Nonlinear Science and Numerical Simulation*, 19(4):1213 – 1228, 2014.

[50] Amir Hossein Gandomi and Amir Hossein Alavi. Krill herd: A new bio-inspired optimization algorithm. *Communications in Nonlinear Science and Numerical Simulation*, 17(12):4831 – 4845, 2012.

[51] Mehdi Neshat, Ghodrat Sepidnam, Mehdi Sargolzaei, and Adel Najaran Toosi. Artificial fish swarm algorithm: a survey of the state-of-the-art, hybridization, combinatorial and indicative applications. *Artificial Intelligence Review*, 42(4):965–997, 2014.

[52] Yang Shiqin, Jiang Jianjun, and Yan Guangxing. A dolphin partner optimization. *Proceedings of the WRI Global Congress on Intelligent Systems*, pages 124–128, 2009.

[53] C. Yang, X. Tu, and J. Chen. Algorithm of marriage in honey bees optimization based on the wolf pack search. *International Conference on Intelligent Pervasive Computing*, pages 462–467, 2007.

[54] Xueyan Lu and Yongquan Zhou. A novel global convergence algorithm: Bee collecting pollen algorithm. *4th International Conference on Intelligent Computing, Springer*, pages 518–525, 2008.

[55] Roth Martin and Wicker Stephen. Termite: A swarm intelligent routing algorithm for mobilewireless ad-hoc networks. In *Stigmergic optimization*, pages 155–184. Springer, 2006.

[56] Pedro C Pinto, Thomas A Runkler, and Joao MC Sousa. Wasp swarm algorithm for dynamic max-sat problems. In *International Conference on Adaptive and Natural Computing Algorithms*, pages 350–357. Springer, 2007.

[57] H. A. Abbass. Mbo: marriage in honey bees optimization-a haplometrosis polygynous swarming approach. In *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*, volume 1, pages 207–214 vol. 1, May 2001.

[58] A. Slowik and H. Kwasnicka. Nature inspired methods and their industry applications - swarm intelligence algorithms. *IEEE Transactions on Industrial Informatics*, PP(99):1–1, 2017.

[59] Xin-She Yang. A new metaheuristic bat-inspired algorithm. *Springer Berlin Heidelberg*, pages 65–74, 2010.

[60] Dervis Karaboga and Bahriye Basturk. *Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems*, pages 789–798. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[61] Gaurav Dhiman and Vijay Kumar. Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications. *Advances in Engineering Software*, 114:48 – 70, 2017.

[62] Gaurav Dhiman, Sen Guo, and Satnam Kaur. ED-SHO: a framework for solving non-linear economic load power dispatch problem using spotted hyena optimizer. *Modern Physics Letters A, In press*, 2018.

[63] Gaurav Dhiman and Vijay Kumar. Astrophysics inspired multi-objective approach for automatic clustering and feature selection in real-life environment. *Modern Physics Letters b*, page 1850385, 2018.

[64] X. S. Yang and Suash Deb. Cuckoo search via levy flights. In *World congress on nature biologically inspired computing*, pages 210–214, 2009.

[65] J. Hoyo, A. Elliott, and J. Sargatal. Handbook of the birds of the world. *Lynx Edicions*, 3:572–599, 1996.

[66] S. M. Macdonald and C.F. Mason. Predation of migrant birds by gulls. *Brit. Birds*, 66:361–363, 1973.

[67] Q Chen, B Liu, Q Zhang, J Liang, P Suganthan, and BY Qu. Problem definitions and evaluation criteria for cec 2015 special session on bound constrained single-objective computationally expensive numerical optimization. *Technical Report, Nanyang Technological University*, 2014.

[68] Seyedali Mirjalili. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowledge-Based Systems*, 89:228 – 249, 2015.

[69] Seyedali Mirjalili, Seyed Mohammad Mirjalili, and Abdolreza Hatamlou. Multi-verse optimizer: A nature-inspired algorithm for global optimization. *Neural Comput. Appl.*, 27(2):495–513, February 2016.

[70] Seyedali Mirjalili. Sca: A sine cosine algorithm for solving optimization problems. *Knowledge-Based Systems*, 96:120 – 133, 2016.

[71] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. Swarm intelligence: From natural to artificial systems. *Oxford University Press, Inc.*, 1999.

[72] F. van den Bergh and A.P. Engelbrecht. A study of particle swarm optimization particle trajectories. *Information Sciences*, 176(8):937 – 971, 2006.

[73] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

[74] H. B. Mann and D. R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *Ann. Math. Statist.*, 18(1):50–60, 03 1947.

[75] Carlos A Coello Coello. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, 191(11-12):1245 – 1287, 2002.

[76] Toshihiko Baba. Slow light in photonic crystals. *Nature photonics*, 2(8):465–473, 2008.

[77] Y. Zhai, H. Tian, and Y. Ji. Slow light property improvement and optical buffer capability in ring-shape-hole photonic crystal waveguide. *Journal of Lightwave Technology*, 29(20):3083–3090, Oct 2011.

[78] Seyed Mohammad Mirjalili, Kambiz Abedi, and Seyedali Mirjalili. Optical buffer performance enhancement using particle swarm optimization in ring-shape-hole photonic crystal waveguide. *Optik - International Journal for Light and Electron Optics*, 124(23):5989 – 5993, 2013.

[79] Jun Wu, Yanping Li, Chao Peng, and Ziyu Wang. Wideband and low dispersion slow light in slotted photonic crystal waveguide. *Optics Communications*, 283(14):2815–2819, 2010.

[80] BK Kannan and Steven N Kramer. An augmented lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. *Journal of mechanical design*, 116(2):405–411, 1994.

[81] Amir Hossein Gandomi and Xin-She Yang. Benchmark problems in structural optimization. *Springer Berlin Heidelberg*, pages 259–281, 2011.

[82] Efrén Mezura-Montes and Carlos A. Coello Coello. Useful infeasible solutions in engineering optimization with evolutionary algorithms. *Springer Berlin Heidelberg*, pages 652–662, 2005.

[83] Charles V. Camp Barron J. Bichon. Design of space trusses using ant colony optimization. *Journal of Structural Engineering*, 130(5):741–751, 2004.

[84] J.F. Schutte and A.A. Groenwold. Sizing design of truss structures using particle swarms. *Structural and Multidisciplinary Optimization*, 25(4):261–269, 2003.

[85] Ali Kaveh and Siamak Talatahari. Optimal design of skeletal structures via the charged system search algorithm. *Structural and Multidisciplinary Optimization*, 41(6):893–911, 2010.

[86] A. Kaveh and S. Talatahari. Size optimization of space trusses using big bang-big crunch algorithm. *Computers and Structures*, 87(17-18):1129 – 1140, 2009.