Knowledge-Based Systems 159 (2018) 20–50

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Emperor penguin optimizer: A bio-inspired algorithm for engineering problems

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

[a] Computer Science and Engineering Department, Thapar Institute of Engineering and Technology, Patiala, Punjab, INDIA

ARTICLE INFO

Keywords:
Optimization techniques
Metaheuristics
Constrained optimization
Unconstrained optimization
Benchmark test functions

ABSTRACT

This paper proposes a novel optimization algorithm, called Emperor Penguin Optimizer (EPO), which mimics the huddling behavior of emperor penguins (*Aptenodytes forsteri*). The main steps of EPO are to generate the huddle boundary, compute temperature around the huddle, calculate the distance, and find the effective mover. These steps are mathematically modeled and implemented on 44 well-known benchmark test functions. It is compared with eight state-of-the-art optimization algorithms. The paper also considers for solving six real-life constrained and one unconstrained engineering design problems. The convergence and computational complexity are also analyzed to ensure the applicability of proposed algorithm. The experimental results show that the proposed algorithm is able to provide better results as compared to the other well-known metaheuristic algorithms.

1. Introduction

During the last few decades, various algorithms have been proposed to solve the variety of real-life engineering optimization problems [1,2]. These optimization problems are very complex in nature because they have more than one local optimum solution. These problems are divided into various categories whether they are constrained or unconstrained, discrete or continuous, static or dynamic, single or multi-objective.

In order to increase the efficiency and accuracy of these problems, researchers have encouraged to rely on metaheuristic optimization algorithms [3]. Metaheuristics become more popular in various field because they do not require gradient information, easy to implement, and bypass the local optima problem.

Generally, metaheuristics are divided into two categories such as single-solution and multiple-solution based problems.

In single-solution based algorithms the searching process starts with one candidate solution whereas in multiple-solution based algorithm the optimization performs using a set of solutions (i.e., population). Multiple-solution or population based metaheuristics have advantages over single-solution based metaheuristics. These are as follows:

- The searching process starts with random generated population i.e, a set of multiple solutions.
- The multiple solutions can share the information between each other around the search space and avoid local optimal solutions.
- The exploration capability of multiple-solution or population based metaheuristics have better than the single-solution based techniques.

Multiple-solution based metaheuristic algorithms are further classified into three categories such as evolutionary-based, physics-based, and swarm-based methods. The first category is generic population based metaheuristic which is inspired from biological evolution i.e., mutation, recombination, and selection. These methods do not make any assumptions about the basic fitness landscape.

The second category is physics-based algorithms in which each search agent can communicate and move throughout the search space according to some physics rules such as gravitational force, electromagnetic force, inertia force, and many more.

The last category is swarm-based algorithms which are inspired by the collective behavior of social creatures. This collective intelligence is based on the interaction of swarm with each other. Swarm-based algorithms are easier to implement than the evolutionary-based algorithms due to include the less number of operators (i.e., selection, crossover, mutation). Apart from this, there are some advantages of swarm-based algorithms which are as follows:

- Swarm-based algorithms can maintain the information about the search space during course of iterations whereas evolutionary-based algorithms can eliminate the information of the previous generations.
- Swarm-based algorithms have few input parameters as compared to the evolutionary techniques.

* Corresponding author.
E-mail addresses: gaurav.dhiman@thapar.edu (G. Dhiman), vijaykumarchahar@gmail.com (V. Kumar).

https://doi.org/10.1016/j.knosys.2018.06.001
Received 16 October 2017; Received in revised form 30 May 2018; Accepted 1 June 2018
Available online 15 June 2018
0950-7051/ © 2018 Elsevier B.V. All rights reserved.

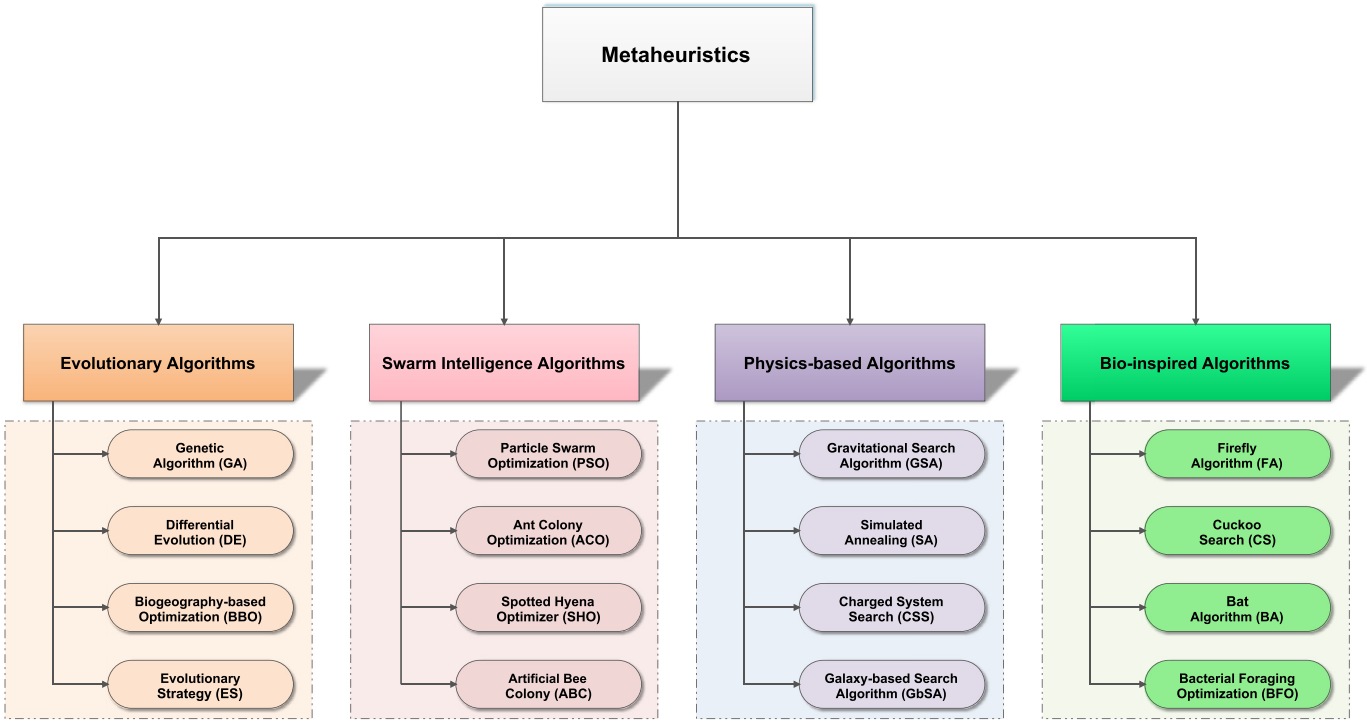

Fig. 1. Classification of metaheuristic algorithms.

- Swarm-based algorithms utilize less memory space for saving the best optimal solutions.

The key phases of metaheuristic algorithms are exploration and exploitation [4,5]. The exploration phase ensures that algorithm investigates the different promising regions in a given search space whereas exploitation ensures the searching of optimal solutions around the promising regions which is obtained in the exploration phase [6]. However, it is difficult to balance between these phases due to its stochastic nature. Therefore, the fine tuning of these two phases is required to achieve the near optimal solutions for a given optimization problem.

This is the one fact which can motivates us to develop a novel metaheuristic algorithm for solving real-life optimization problems. Another fact of our motivation is the set of given problem in which the performance of one optimizer does not guarantee to solve other optimization problem with different nature [7].

This paper presents a novel bio-inspired metaheuristic algorithm named as Emperor Penguin Optimizer (EPO) for optimizing the both constrained and unconstrained optimization problems. Emperor Penguin Optimizer (EPO) is inspired by social huddling behavior of emperor penguins to survive successfully in the depth of Antarctic winter. The main steps of EPO are inspired by huddling behavior which computes the huddling boundary, temperature, distance, and effective mover around the huddle. The performance of EPO algorithm has been evaluated on forty-four linear and non-linear benchmark test functions. The proposed algorithm has also been tested on seven non-linear and mixed integer structural optimization problems.

The rest of this paper is structured as follows: Section 2 presents the related works of optimization problems. The proposed EPO algorithm is discussed in Section 3. The experimental results and discussion is presented in Section 4. Section 5 focuses on the applications of EPO in engineering problems. Finally, the conclusion and some future research directions are given in Section 6.

2. Background

This section firstly describes the basic concepts of optimization. The related works of various metaheuristics are then discussed. Finally, the motivation of this work is provided.

2.1. Preliminaries

Many real-life problems need to achieve several objectives such as minimize risks, minimize cost, maximize reliability, etc. The main aim of single-objective optimization is to find the best optimal solution and deals with only one objective which is to be minimized or maximized. The mathematical formulation of single-objective optimization is as follows:

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

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, g_j is the jth inequality constraints, q is the number of equality constraints, h_j is the jth equality constraints, r is the number of variables, lb_j is the lower bound and ub_j is the upper bound of the jth variable.

Therefore, the presence of local solutions in a given search region is the main challenging task when solving optimization problems. In single-objective optimization, there is only one best solution i.e., the best objective value. However, there are many other solutions called as local solutions, which may be closer than the obtained objective value. Since these solutions are not considered as the global solution which may causes the entrapment due to the presence of these local solutions. It may occurs the stagnation problem due to assumes the local solutions as the global optimum. Hence, an optimization algorithm should be able to avoid such problem and determine the global optimum solution

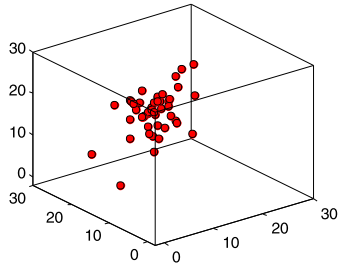

Fig. 2. Generate huddle to conserve energy and movement of search agents towards the best neighbor.

efficiently.

The convergence is an another main challenge for any optimization algorithm. It is not necessary that an algorithm which is able to avoid local solutions, provides better convergence towards the global optima.

Therefore, these two conflict trade-offs are the main challenges of an optimization algorithm to solve real-life problems.

2.2. Related works

Population based metaheuristic algorithms are classified into three categories namely Physics-based, Evolutionary-based, and Swarm-based methods. The broadly classification of metaheuristic optimization algorithms is shown in Fig. 1.

Physics-based metaheuristic algorithms utilize the concepts of physics such as electromagnetic force, inertia force, and gravitational force, for the information sharing among search agents in the given search space. The general mechanism of these algorithms is different from the other approaches due to the strategy of search agents as per physics rules. The well-known physics-based metaheuristic algorithms are Simulated Annealing (SA) [8] and Gravitational Search Algorithm (GSA) [9]. 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. SA is advantageous in terms to deal with non-linear models and noisy data. The main advantage of SA over other search methods is its ability to search the global optimal solution. However, it suffers from high computational time especially if the fitness function is very complex and non-linear in nature. 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. GSA requires only two input parameters to adjust i.e., mass and velocity. It is easy to implement. The ability to find near the global optimum solution makes the GSA differ from other optimization algorithms. However, it suffers from computational time and convergence problem if the initial population is not generated well.

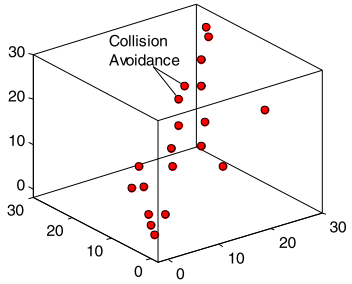

Fig. 4. Collision avoidance between search agents.

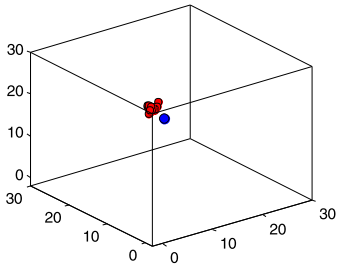

Fig. 5. Updated positions of emperor penguins towards the best search agent.

Some of the other popular algorithms are: Big-Bang Big-Crunch (BBBC) [10], Charged System Search (CSS) [11], Black Hole (BH) [12] algorithm, Central Force Optimization (CFO) [13], Small-World Optimization Algorithm (SWOA) [14], Artificial Chemical Reaction Optimization Algorithm (ACROA) [15], Ray Optimization (RO) algorithm [16], Galaxy-based Search Algorithm (GbSA) [17], and Curved Space Optimization (CSO) [18].

The second subclass of metaheuristic algorithms is evolutionary-based algorithms. These algorithms are inspired by theory of natural selection and biological evolution. These algorithms 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) [19]. The evolution starts with the randomly generated individuals from a population. The fitness of each individual is computed in each generation. The crossover and mutation operators are applied on individual to create a new population. The best individuals can generate the new population during the course of iterations. However, compared to other stochastic methods genetic algorithm has the advantage that it can be parallelized with little effort and not necessarily remain trapped in a sub-optimal local maximum or minimum of the target function. GA may provides local minima of a function that can steer the search in the wrong direction for some of the

Fig. 3. Huddling behavior of emperor penguins.

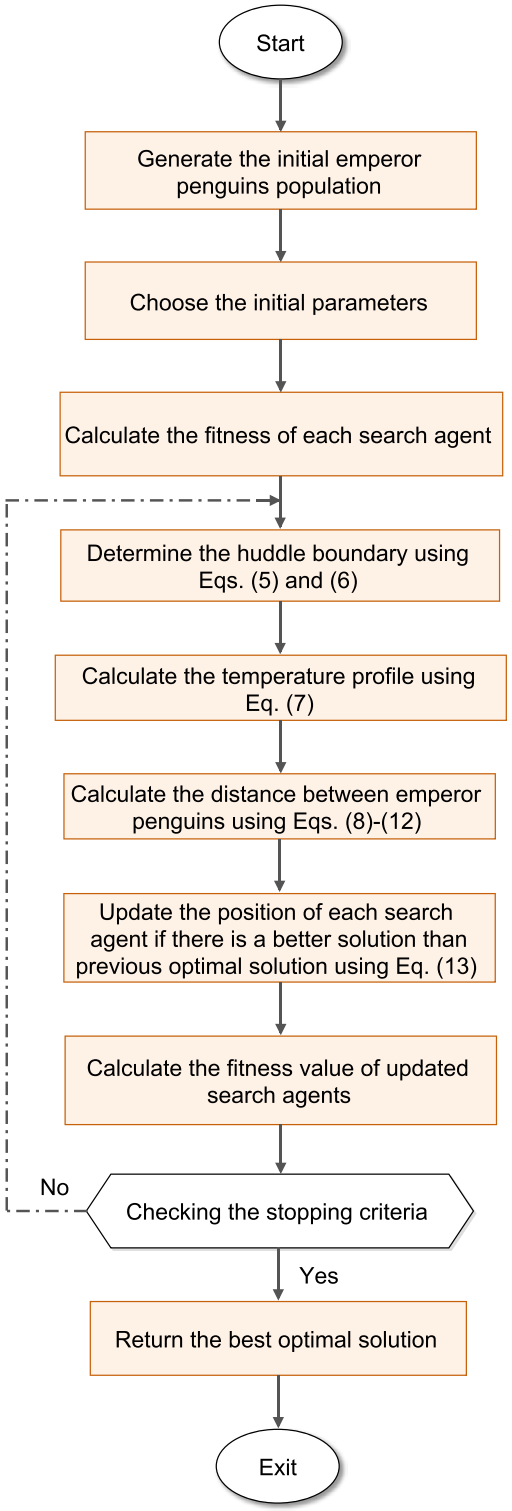

Fig. 6. Flowchart of the proposed EPO algorithm.

optimization problems.

Differential Evolution (DE) [20] is another evolutionary-based metaheuristic algorithm that optimizes a problem by maintaining a candidate solutions and creates new candidate solutions by combining the existing ones. It can keep the candidate solution which has the best fitness value for optimization problem. It has an ability to handle the non-differentiable and non-linear cost functions. There are only few parameters to steer the minimization problem. The parameter tuning is a main challenge in DE because same parameters may not guarantee the global optimum solution.

Apart from these, some of the other popular evolutionary-based algorithms are Genetic Programming (GP) [21], Evolution Strategy (ES) [22], and Biogeography-based Optimizer (BBO) [23].

The third main category is swarm-based metaheuristic algorithms which are based on the collective behavior of social creatures. These algorithms are generally inspired from natural colonies, flock, herds, and so on. The most popular algorithm is Particle Swarm Optimization (PSO) which was proposed by Kennedy and Eberhart [24]. In PSO, particles move around the search space using the combination of best solutions [25]. The whole process is repeated until the termination criterion is satisfied. The main advantage of PSO is that it has no overlapping and mutation calculation. During simulation, the most optimist particle can transmit information among the other particles. However, it suffers from the stagnation problem.

Ant Colony Optimization (ACO) is another popular swarm intelligence algorithm which was proposed by Dorigo et al. [26]. The main inspiration behind this algorithm is the social behavior of ants in ant colony. The social intelligence of ants is to find the shortest path between the source food and nest. ACO is able to solve the travelling salesman and similar problems in an efficient way that can be advantageous of ACO over other approaches. The theoretical analysis of a problem is very difficult using ACO because the computational cost is high during convergence.

Bat-inspired Algorithm (BA) [27] is inspired by the echolocation behavior of bats. The another well-known swarm-based metaheuristic is Artificial Bee Colony (ABC) algorithm [28] which is inspired by the collective behavior of bees to find the food sources. Spotted Hyena Optimizer (SHO) [29] is a recently develop bio-inspired metaheuristic algorithm that mimics the searching, hunting, and attacking behaviors of spotted hyenas in nature. The main concept behind this technique is the social relationship and collective behavior of spotted hyenas for hunting strategy. Cuckoo Search (CS) [30] 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.

The other well-known metaheuristic algorithms are Monkey Search [31], Bacterial Foraging Optimization Algorithm [32], Firefly Algorithm (FA) [33], Fruit fly Optimization Algorithm (FOA) [34–37], Golden section line search algorithm [38], Fibonacci search method [39], Bird Mating Optimizer (BMO) [40], Krill Herd (KH) [41], Artificial Fish-Swarm Algorithm (AFSA) [42], Dolphin Partner Optimization (DPO) [43], Hunting Search (HS) [44], and Moth-flame Optimization Algorithm (MFO) [45].

2.3. Motivation

It has been observed from the literature that multiple-solution or population based algorithms are able for solving real-life optimization problems [46–48]. They are able to avoid local optima problem, exploring throughout the search space, and exploit the global optimum. However, population based techniques are more reliable than single-solution based techniques because of more function evaluations [49].

According to No Free Lunch theorem [7], there is no optimization algorithm which is able to solve all optimization problems. This fact will attract the researchers of different fields to propose a new optimization algorithm. Therefore, these two reasons are the main motivation of this research work to propose a new population based metaheuristic algorithm.

3. Emperor Penguin Optimizer (EPO)

In this section, the inspiration and mathematical modeling of the proposed algorithm are described in detail.

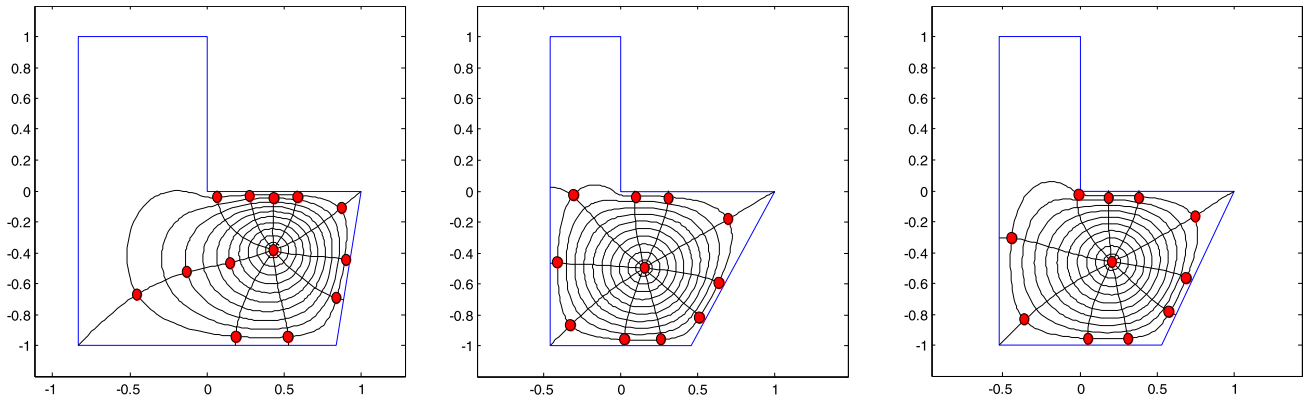

Fig. 7. L-shaped polygon huddling boundary of emperor penguins.

3.1. Inspiration

The emperor penguin, scientifically named as *Aptenodytes forsteri*, is the tallest and heaviest in all of the penguin species. The male and female emperor penguins are very much similar in terms of plumage and size. The dorsal side and head are black with white belly, pale yellow breast, and bright yellow ear patches.

Emperor penguins spend their lives in open ice and breeds during winter season. During the breeding season, emperor penguins come ashore in huge colonies which includes hundreds of thousands of emperor penguins. Females lay a single egg and can travel 50 miles to reach the ocean for hunting. The emperor penguin is a social like animal by its foraging behavior and hunting together in a group.

In sea, emperor penguins can dive up to 1900 feet deeper and stay under sea for more than 25 min. Emperor penguins are flightless like other penguins species with stiffened and flattened wings.

Emperor penguins are the only species that huddles to survive during the Antarctic winter. The huddling behavior of emperor penguins is decomposed into four phases [50]:

- Generate and determine the huddle boundary of emperor penguins.
- Calculate the temperature profile around the huddle.
- Determine the distance between emperor penguins.
- Relocate the effective mover.

An important feature of this huddling behavior is that each penguin has an equal opportunity to the warmth of huddle.

Fig. 3 shows the huddling behavior of emperor penguins. In this figure, the emperor penguin (X^*, Y^*) can update its position towards the positions of other emperor penguins and reaches different number of places about the current position.

3.2. Mathematical model and optimization algorithm

In this section, the huddling behavior of emperor penguins is mathematically modeled. The main aim of this model is to find an effective mover. The huddle is assumed to be situated on two dimensional L-shape polygon plane. Firstly, emperor penguins generate the huddle boundary randomly. Thereafter, the temperature profile around the huddle is computed. The distance between emperor penguins is also calculated which will be helpful for more exploration and exploitation. Finally, the effective mover i.e., the best optimal solution is obtained and recompute the boundary of huddle with updated positions of emperor penguins (or search agents). These steps are explained in the preceding sections.

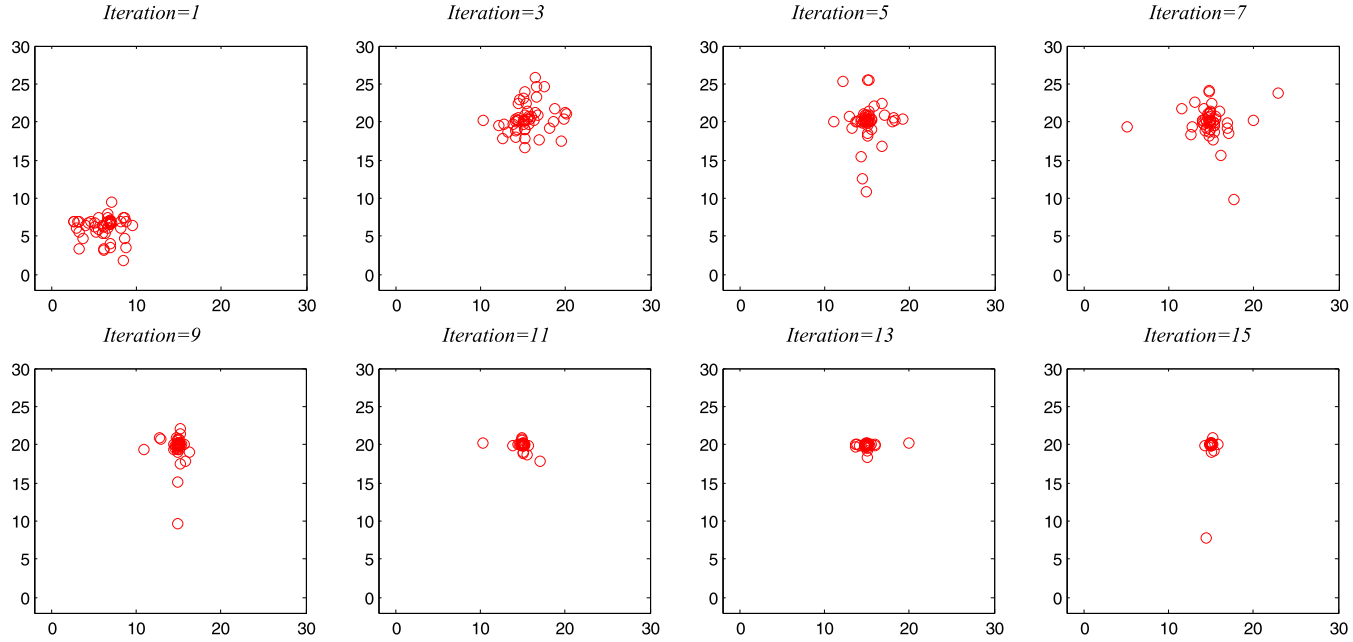

Fig. 8. Swarm behaviors of emperor penguins in two-dimensional environment.

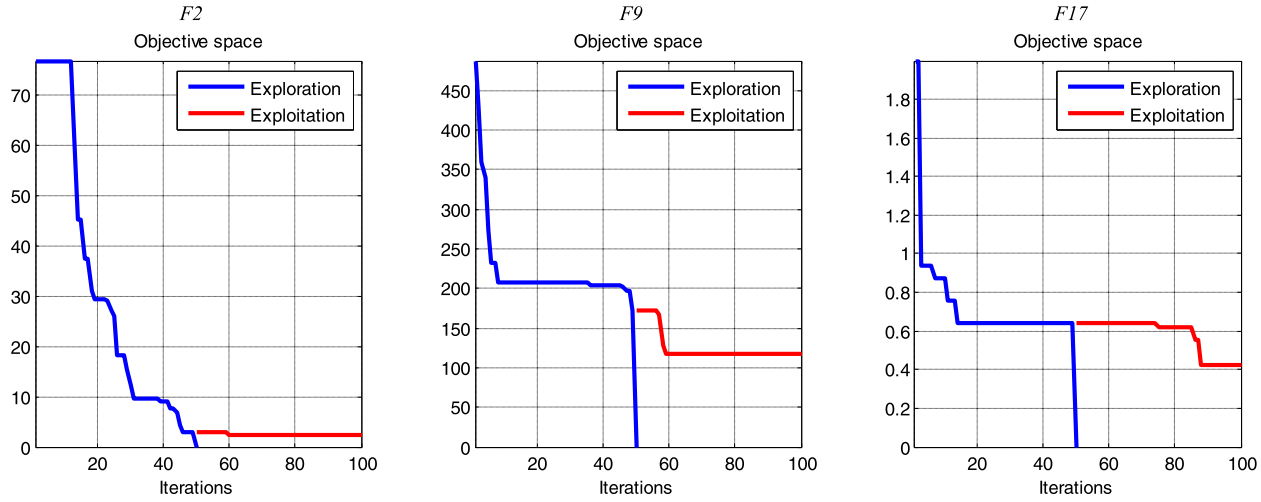

Fig. 9. Exploration and Exploitation phases of EPO on F_2, F_9, and F_{17} benchmark test functions.

3.2.1. Generate and determine the huddle boundary

During huddling, emperor penguins generally position themselves on a polygon shape grid boundary. The emperor penguins have at least two neighbors which are chosen randomly in the huddle as shown in Fig. 3. The wind flow around the huddle is determined to find the huddle boundary around a polygon. However, the wind flow is faster than the movement of a emperor penguin. The concepts of complex variables is used to describe the randomly generated huddle boundary of emperor penguin.

Let Φ defines the wind velocity and Ψ be the gradient of Φ.

$$\Psi = \nabla\Phi \tag{5}$$

Vector Ω is combined with Φ to generate the complex potential.

$$F = \Phi + i\Omega, \tag{6}$$

where i denotes the imaginary constant and F is an analytical function on the polygon plane. Fig. 3 shows the effect of Eq. (6) in two-dimensional environment. In this figure, the emperor penguins can update their position randomly towards the position of emperor penguin which is to be situated at the center of L-shaped polygon region with highest effective fitness rate during iteration process.

3.2.2. Temperature profile around the huddle

The emperor penguins generate huddle to conserve energy and maximize the ambient temperature in the huddle (see Fig. 2). To mathematically model this situation, we assume that the temperature $T = 0$ when the radius of polygon $R > 1$ and temperature $T = 1$ when the radius becomes $R < 1$. This temperature profile is responsible for exploration and exploitation process for emperor penguins with different locations. The temperature profile around the huddle T' is computed as follows:

$$T' = \left(T - \frac{Max_{iteration}}{x - Max_{iteration}}\right)$$
$$T = \begin{cases} 0, & \text{if } R > 1 \\ 1, & \text{if } R < 1, \end{cases} \tag{7}$$

where x defines the current iteration, $Max_{iteration}$ denotes the maximum number of iterations, R is the radius, and T is the time for finding best optimal solution in a search space.

3.2.3. Distance between emperor penguins

The distance between emperor penguin and best obtained optimal solution is computed after the generation of huddle boundary. The current best optimal solution is the solution whose fitness value is close to the optimum. The other search agents (or emperor penguins) will

Table 1
Parameter settings for algorithms.

#	Algorithms	Parameters	Values
1.	Emperor Penguin Optimizer (EPO)	Search Agents	80
		Temperature Profile (T')	[1, 1000]
		$\overrightarrow{A}$ Constant	[−1.5, 1.5]
		Function $S()$	[0, 1.5]
		Parameter M	2
		Parameter f	[2, 3]
		Parameter l	[1.5, 2]
		Number of Generations	1000
2.	Spotted Hyena Optimizer (SHO)	Search Agents	80
		Control Parameter ($\overrightarrow{h}$)	[5, 0]
		$\overrightarrow{M}$ Constant	[0.5, 1]
		Number of Generations	1000
3.	Grey Wolf Optimizer (GWO)	Search Agents	80
		Control Parameter ($\overrightarrow{a}$)	[2, 0]
		Number of Generations	1000
4.	Particle Swarm Optimization (PSO)	Number of Particles	80
		Inertia Coefficient	0.75
		Cognitive and Social Coeff	1.8, 2
		Number of Generations	1000
5.	Multi-Verse Optimizer (MVO)	Search Agents	80
		Wormhole Existence Prob.	[0.2, 1]
		Travelling Distance Rate	[0.6, 1]
		Number of Generations	1000
6.	Sine Cosine Algorithm (SCA)	Search Agents	80
		Number of Elites	2
		Number of Generations	1000
7.	Gravitational Search Algorithm (GSA)	Search Agents	80
		Gravitational Constant	100
		Alpha Coefficient	20
		Number of Generations	1000
8.	Genetic Algorithm (GA)	Population Size	80
		Crossover and Mutation	0.9, 0.05
		Number of Generations	1000
9.	Harmony Search (HS)	Harmony Memory	80
		Harmony Rate	0.95
		Neighbouring Value Rate	0.30
		Discrete Set	17700
		Fret Width	1
		Number of Generations	1000

Table 2
The obtained optimal values on unimodal, multimodal, fixed-dimension multimodal, composite, and CEC 2015 benchmark test functions using different simulation runs (i.e., 100, 500, 800, and 1000).

	Functions						
Iterations	F_1	F_5	F_{11}	F_{17}	F_{23}	F_{24}	$CEC-1$
100	2.71E−17	7.07E+00	3.21E−03	9.07E−01	−2.11E+00	3.53E+02	2.60E+06
500	5.44E−20	6.01E+00	1.20E−03	7.97E−01	−2.97E+00	3.15E+02	2.56E+05
800	4.00E−25	5.52E+00	7.22E−04	6.12E−01	−3.11E+00	2.95E+02	2.01E+05
1000	**2.42E−29**	**5.04E+00**	**4.20E−05**	**3.88E−01**	**−3.47E+00**	**2.35E+02**	**1.50E+05**

update their positions according to current best optimal solution which is mathematically defined as follows:

$$\overrightarrow{D_{ep}} = Abs\left(S(\overrightarrow{A})\cdot\overrightarrow{P(x)} - \overrightarrow{C}\cdot\overrightarrow{P_{ep}(x)}\right), \tag{8}$$

where $\overrightarrow{D_{ep}}$ represents the distance between the emperor penguin and best fittest search agent (i.e., best emperor penguin whose fitness value is less), x indicates the current iteration. $\overrightarrow{A}$ and $\overrightarrow{C}$ are used to avoid the collision between neighbors (i.e., other emperor penguins) as shown in Fig. 4. $\overrightarrow{P}$ defines the best optimal solution (i.e., fittest emperor penguin), $\overrightarrow{P_{ep}}$ indicates the position vector of emperor penguin. $S()$ defines the social forces of emperor penguins that is responsible to move towards the direction of best optimal search agent. The vectors $\overrightarrow{A}$ and $\overrightarrow{C}$ are computed as follows:

$$\overrightarrow{A} = (M \times (T' + P_{grid}(Accuracy)) \times Rand()) - T' \tag{9}$$

$$P_{grid}(Accuracy) = Abs(\overrightarrow{P} - \overrightarrow{P_{ep}}) \tag{10}$$

$$\overrightarrow{C} = Rand(), \tag{11}$$

where M is the movement parameter that maintains a gap between search agents for collision avoidance. The value of parameter M is set to 2. T' is the temperature profile around the huddle, $P_{grid}(Accuracy)$ defines the polygon grid accuracy by comparing the difference between emperor penguins, and $Rand()$ is a random function lies in the range of [0, 1].

The function $S()$ is calculated as follows:

$$S(\overrightarrow{A}) = \left(\sqrt{f\cdot e^{-x/l} - e^{-x}}\right)^2, \tag{12}$$

where e defines the expression function. f and l are control parameters for better exploration and exploitation. The values of f and l lie in the range of [2, 3] and [1.5, 2], respectively. Note that it has been observed that the proposed algorithm provides better results between these ranges. The sensitivity analysis of these parameters is described in Section 4.6.

3.2.4. Relocate the mover

The positions of emperor penguins are updated according to the best obtained optimal solution i.e., mover (see Fig. 5). This mover is responsible for changing the positions of other search agents in a given search space and vacates its current position. The following equation is proposed to update the next position of an emperor penguin:

$$\overrightarrow{P_{ep}}(x+1) = \overrightarrow{P(x)} - \overrightarrow{A}\cdot\overrightarrow{D_{ep}}, \tag{13}$$

where $\overrightarrow{P_{ep}}(x+1)$ represents the next updated position of emperor penguin. During iteration process, the huddling behavior of emperor penguins is recomputed once the mover has been re-located.

The pseudo code of EPO algorithm is shown in Algorithm. There are some interesting points about the proposed EPO algorithm which are given below:

- The proposed algorithm saves the best solutions which is obtained so far during the course of iterations.
- The proposed polygon grid mechanism defines a L-shaped polygon grid around the solutions which can be extended to higher dimensions as shown in Fig. 7.
- $\overrightarrow{A}$ and $\overrightarrow{C}$ assist the candidate solutions to behave more randomly in a search space and responsible to avoid the collisions between search agents.
- The proposed distance method allows candidate solutions to locate the possibly position of the best fittest emperor penguin.
- The convergence behaviors of common optimization algorithms suggests that the exploitation tends to increase the speed of convergence, while exploration tends to decrease the convergence rate of the algorithm. Therefore, the possibility of better exploration and exploitation is done by the adjusted values of vectors $\overrightarrow{A}$ and $\overrightarrow{C}$. These behaviors are shown in Fig. 9 which shows that half of iterations are responsible for exploration and the other half are responsible for exploitation in a given search space.
- The swarm behavior of emperor penguins in a search region defines the effectively collective behavior of EPO algorithm as shown in Fig. 8.

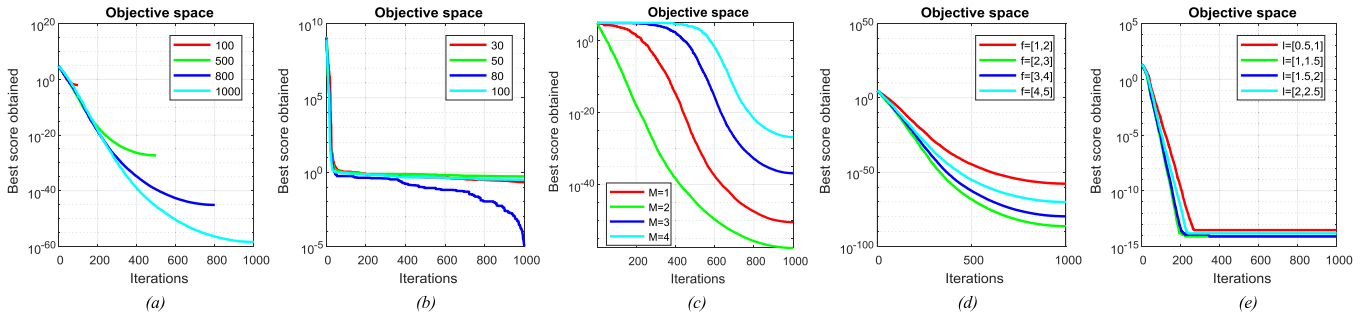

Fig. 10. Sensitivity analysis of proposed EPO algorithm for, (a) Number of iterations; (b) Number of search agents; (c) Control parameter M; (d) Control parameter f; (e) Control parameter l.

Input: the emperor penguins population $\vec{P_{ep}}(x)$ $(x \leftarrow 1, 2, \ldots, n)$
Output: the best obtained search agent $\vec{P}$

procedure EPO
Initialize the parameters $T', \vec{A}, \vec{C}, S(), R$, and $Max_{iteration}$
  while $(x < Max_{Iteration})$ **do**
    FITNESS(P_{ep}) /* Compute the fitness of each search agent using **FITNESS** function*/
    $R \leftarrow Rand()$ /* Generate random number in range [0, 1] */
    if$(R > 1)$**then**
      $T \leftarrow 0$
    else
      $T \leftarrow 1$
    end if
    $T' \leftarrow \left(T - \frac{Max_{iteration}}{x - Max_{iteration}}\right)$ /* Compute the temperature profile around the huddle */
    for i ← 1 to n **do**
      for j ← 1 to n **do**
        Compute the vectors $\vec{A}$ and $\vec{C}$ using Eqs. (9)–(11)
        Compute the function $S(\vec{A})$ using Eq. (12)
        Update the position of current agent using Eq. (13)
      end for
    end for
    Update parameters $T', \vec{A}, \vec{C}$, and $S()$
    Amend search agent which goes beyond the region of search space
    FITNESS(P_{ep}) /* Again compute the fitness value of updated search agents using **FITNESS** function*/
    Update $\vec{P}$ if there is a better solution than previous optimal solution i.e., (FIT_{best})
    $x \leftarrow x + 1$
  end while
return $\vec{P}$
end procedure

procedure FITNESS(P_{ep})
  for i ← 1 to n **do**
    $FIT[i] \leftarrow FITNESS\_FUNCTION(P_{ep})$ /* Compute the fitness of each individual */
  end for
  $FIT_{best} \leftarrow$ **BEST**$(FIT[])$ /* Compute the best fitness value using **BEST** function */
return FIT_{best}
end procedure

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

Algorithm :. Emperor Penguin Optimizer.

3.3. Steps and flowchart of proposed EPO algorithm

The steps and flowchart (see Fig. 6) of proposed EPO are summarized as follows:

Step 1: Initialize the emperor penguins population $\overrightarrow{P_{ep}(x)}$, where $x = 1, 2, \ldots, n$.
Step 2: Choose the initial parameters: T', $\overrightarrow{A}$, $\overrightarrow{C}$, $S()$, R, and $Max_{iteration}$.
Step 3: Now, calculate the fitness value of each search agent.
Step 4: Determine the huddle boundary of emperor penguins using Eqs. (5) and (6).
Step 5: Calculate the temperature profile T' around the huddle using Eq. (7).
Step 6: Compute the distance between the emperor penguins using Eqs. (8)–(12).
Step 7: Update the positions of other search agents using Eq. (13).

Table 3
The obtained optimal values on unimodal, multimodal, fixed-dimension multimodal, composite, and CEC 2015 benchmark test functions, where number of iterations is fixed as 1000. The number of search agents are varied from 30 to 100.

	Functions						
Search agents	F_1	F_5	F_{11}	F_{17}	F_{23}	F_{24}	$CEC-1$
30	1.51E−16	7.01E+00	3.11E−03	5.27E−01	−2.53E+00	1.33E+02	2.86E+06
50	5.43E−19	6.31E+00	2.30E−03	7.20E−01	−2.80E+00	3.20E+02	2.22E+05
80	**2.22E−28**	**5.00E+00**	**4.24E−05**	**3.22E−01**	**−3.49E+00**	**2.29E+02**	**1.47E+05**
100	5.97E−23	5.42E+00	8.32E−04	6.18E−01	−3.13E+00	2.96E+02	2.18E+05

Table 4
The obtained optimal values on unimodal, multimodal, fixed-dimension multimodal, composite, and CEC 2015 benchmark test functions, where number of iterations and search agents are fixed as 1000 and 80, respectively. The parameter M is varied from [1, 2, 3, 4].

	Functions						
l	F_1	F_5	F_{11}	F_{17}	F_{23}	F_{24}	$CEC-1$
1	4.41E−17	6.10E+00	2.41E−04	5.76E−01	−2.21E+00	3.36E+02	2.53E+06
2	**4.00E−22**	**5.13E+00**	**1.19E−06**	**4.87E−02**	**−3.50E+00**	**2.24E+02**	**1.47E+05**
3	7.48E−17	7.83E+00	2.40E−03	5.26E−01	−2.93E+00	4.02E+02	3.43E+05
4	1.76E−19	7.02E+00	4.26E−04	5.49E−01	−3.08E+00	2.76E+02	2.19E+05

Table 5
The obtained optimal values on unimodal, multimodal, fixed-dimension multimodal, composite, and CEC 2015 benchmark test functions, where number of iterations and search agents are fixed as 1000 and 80, respectively. The parameter f is varied from [1, 2] to [4, 5].

	Functions						
f	F_1	F_5	F_{11}	F_{17}	F_{23}	F_{24}	$CEC-1$
[1, 2]	6.23E−18	7.05E+00	3.34E−03	8.79E−01	−2.14E+00	3.47E+02	2.53E+06
[2,3]	**2.22E−29**	**5.03E+00**	**4.22E−05**	**3.85E−01**	**−3.50E+00**	**2.33E+02**	**1.48E+05**
[3,4]	3.42E−19	6.00E+00	1.17E−03	7.86E−01	−2.89E+00	3.11E+02	2.50E+05
[4,5]	4.00E−23	5.60E+00	7.40E−04	6.10E−01	−3.17E+00	2.93E+02	2.07E+05

Table 6
The obtained optimal values on unimodal, multimodal, fixed-dimension multimodal, composite, and CEC 2015 benchmark test functions, where number of iterations and search agents are fixed as 1000 and 80, respectively. The parameter f is fixed as [2, 3]. The parameter l is varied from [0.5, 1] to [2, 2.5].

	Functions						
l	F_1	F_5	F_{11}	F_{17}	F_{23}	F_{24}	$CEC-1$
[0.5, 1]	5.46E−19	7.18E+00	3.42E−03	8.92E−01	−2.23E+00	3.40E+02	2.55E+06
[1, 1.5]	1.41E−21	5.86E+00	1.34E−03	7.76E−01	−2.94E+00	3.19E+02	2.75E+05
[1.5, 2]	**2.28E−28**	**5.01E+00**	**4.13E−05**	**3.80E−01**	**−3.50E+00**	**2.31E+02**	**1.51E+05**
[2, 2.5]	8.95E−24	5.55E+00	7.43E−04	6.00E−01	−3.07E+00	2.80E+02	2.00E+05

Step 8: Check whether any search agent goes beyond the boundary in a given search space and then amend it.
Step 9: Calculate the updated search agent fitness value and update the position of previously obtained optimal solution.
Step 10: The algorithm will be stopped until the stopping criterion is satisfied. Otherwise, return to Step 5.
Step 11: Return the best optimal solution, after stopping criteria, which is obtained so far.

3.4. Computational complexity

In this section, the computational complexity of proposed EPO algorithm is discussed. Both the time and space complexities of the proposed algorithm are given below.

3.4.1. Time complexity

1. Population initialization process requires $\mathcal{O}(n \times d)$ time, where n indicates the population size and d indicates the dimension of a given problem.
2. The fitness of each agent requires $\mathcal{O}(Max_{iteration} \times n \times d)$ time, where $Max_{iteration}$ is the maximum number of iteration to simulate the proposed algorithm.
3. The function $S()$ requires $\mathcal{O}(N)$ time where N defines the social forces of emperor penguins for better exploration and exploitation.
4. Steps 2 and 3 is repeated until the termination criteria is satisfied which needs $\mathcal{O}(k)$ time.

Hence, the total complexity of Steps 2 and 3 is $\mathcal{O}(n \times Max_{iteration} \times d \times N)$. Therefore, the overall time complexity of EPO algorithm is $\mathcal{O}(k \times n \times Max_{iteration} \times d \times N)$.

Table 7
Results of unimodal benchmark test functions.

F	EPO		SHO		GWO		PSO		MVO		SCA		GSA		GA		HS	
	Ave	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*
F_1	5.71E-28	8.31E-29	**0.00E+00**	**0.00E+00**	4.61E−23	7.37E−23	4.98E−09	1.40E−08	2.81E−01	1.11E−01	3.55E−02	1.06E−01	1.16E−16	6.10E−17	1.95E−12	2.01E−11	7.86E−10	8.11E−09
F_2	6.20E−40	3.32E−40	**0.00E+00**	**0.00E+00**	1.20E−34	1.30E−34	7.29E−04	1.84E−03	3.96E−01	1.41E−01	3.23E−05	8.57E−05	1.70E−01	9.29E−01	6.53E−18	5.10E−17	5.99E−20	1.11E−17
F_3	2.05E−19	9.17E−20	**0.00E+00**	**0.00E+00**	1.00E−14	4.10E−14	1.40E+01	7.13E+00	4.31E+01	8.97E+00	4.91E+03	3.89E+03	4.16E+02	1.56E+02	7.70E−10	7.36E−09	9.19E−05	6.16E−04
F_4	**4.32E−18**	**3.98E−19**	7.78E−12	8.96E−12	2.02E−14	2.43E−14	6.00E−01	1.72E−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	8.73E−01	1.19E−01
F_5	**5.07E+00**	**4.90E−01**	8.59E+00	5.53E−01	2.79E+01	1.84E+00	4.93E+01	3.89E+01	1.18E+02	1.43E+02	7.37E+02	1.98E+03	3.85E+01	3.47E+01	5.57E+02	4.16E+01	8.91E+02	2.97E+02
F_6	**7.01E−19**	**4.39E−20**	2.46E−01	1.78E−01	6.58E−01	3.38E−01	9.23E−09	1.78E−08	3.15E−01	9.98E−02	4.88E+00	9.75E−01	1.08E−16	4.00E−17	3.15E−01	9.98E−02	8.18E−17	1.70E−18
F_7	**2.71E−05**	**9.26E−06**	3.29E−05	2.43E−05	7.80E−04	3.85E−04	6.92E−02	2.87E−02	2.02E−02	7.43E−03	3.88E−02	5.79E−02	7.68E−01	2.77E+00	6.79E−04	3.29E−03	5.37E−01	1.89E−01

Table 8
Results of multimodal benchmark test functions.

F	EPO		SHO		GWO		PSO		MVO
	Ave	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*
F_8	**−8.76E+02**	5.92E+01	−1.16E+02	**2.72E+01**	−6.14E+02	9.32E+01	−6.01E+02	1.30E+02	−6.92E+02
F_9	6.90E−01	4.81E−01	**0.00E+00**	**0.00E+00**	4.34E−01	1.66E+00	4.72E+01	1.03E+01	1.01E+02
F_{10}	**8.03E−16**	2.74E−14	2.48E+00	1.41E+00	1.63E−14	**3.14E−15**	3.86E−02	2.11E−01	1.15E+00
F_{11}	4.20E−05	4.73E−04	**0.00E+00**	**0.00E+00**	2.29E−03	5.24E−03	5.50E−03	7.39E−03	5.74E−01
F_{12}	5.09E−03	3.75E−03	3.68E−02	1.15E−02	3.93E−02	2.42E−02	**1.05E−10**	**2.06E−10**	1.27E+00
F_{13}	**0.00E+00**	**0.00E+00**	9.29E−01	9.52E−02	4.75E−01	2.38E−01	4.03E−03	5.39E−03	6.60E−02

F	MVO	SCA		GSA		GA		HS	
	Std	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*
F_8	9.19E+01	−3.81E+02	2.83E+01	−2.75E+02	5.72E+01	−5.11E+02	4.37E+01	−4.69E+01	3.94E+01
F_9	1.89E+01	2.23E+01	3.25E+01	3.35E+01	1.19E+01	1.23E−01	4.11E+01	4.85E−02	3.91E+01
F_{10}	7.87E−01	1.55E+01	8.11E+00	8.25E−09	1.90E−09	5.31E−11	1.11E−10	2.83E−08	4.34E−07
F_{11}	1.12E−01	3.01E−01	2.89E−01	8.19E+00	3.70E+00	3.31E−06	4.23E−05	2.49E−05	1.34E−04
F_{12}	1.02E+00	5.21E+01	2.47E+02	2.65E−01	3.14E−01	9.16E−08	4.88E−07	1.34E−05	6.23E−04
F_{13}	4.33E−02	2.81E+02	8.63E+02	5.73E−32	8.95E−32	6.39E−02	4.49E−02	9.94E−08	2.61E−07

Table 9
Results of fixed-dimension multimodal benchmark test functions.

F	EPO		SHO		GWO		PSO		MVO
	Ave	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*
F_{14}	**1.08E+00**	4.11E-02	9.68E+00	3.29E+00	3.71E+00	3.86E+00	2.77E+00	2.32E+00	9.98E+01
F_{15}	**8.21E−03**	**4.09E−03**	9.01E−03	1.06E−03	3.66E−02	7.60E−02	9.09E−03	2.38E−03	7.15E−02
F_{16}	−1.02E+00	9.80E−07	**−1.03E+00**	**2.86E−11**	−1.02E+00	7.02E−09	−1.02E+00	0.00E+00	−1.02E+00
F_{17}	**3.98E−01**	5.39E−05	3.98E−01	2.46E−01	3.98E−01	7.00E−07	3.98E−01	9.03E−16	3.98E−01
F_{18}	**3.00E+00**	1.15E−08	3.00E+00	9.05E+00	3.00E+00	7.16E−06	3.00E+00	6.59E−05	3.00E+00
F_{19}	**−3.86E+00**	6.50E−07	−3.71E+00	4.39E−01	−3.84E+00	1.57E−03	−3.80E+00	**3.37E−15**	−3.77E+00
F_{20}	−2.81E+00	7.11E−01	−1.44E+00	5.47E−01	−3.27E+00	7.27E−02	**−3.32E+00**	2.66E−01	−3.23E+00
F_{21}	−8.07E+00	2.29E+00	−2.08E+00	**3.80E−01**	**−9.65E+00**	1.54E+00	−7.54E+00	2.77E+00	−7.38E+00
F_{22}	**−10.01E+00**	3.97E−02	−1.61E+00	**2.04E−04**	−1.04E+00	2.73E−04	−8.55E+00	3.08E+00	−8.50E+00
F_{23}	−3.41E+00	1.11E−02	−1.68E+00	2.64E−01	**−1.05E+01**	**1.81E−04**	−9.19E+00	2.52E+00	−8.41E+00

F	MVO	SCA		GSA		GA		HS	
	Std	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*
F_{14}	**9.14E−12**	1.26E+00	6.86E−01	3.61E+00	2.96E+00	4.39E+00	4.41E−02	6.79E+00	1.12E+00
F_{15}	1.26E−01	1.01E−02	3.75E−03	6.84E−02	7.37E−02	7.36E−02	2.39E−03	5.15E−02	3.45E−03
F_{16}	4.74E−08	−1.02E+00	3.23E−05	−1.02E+00	0.00E+00	−1.02E+00	4.19E−07	−1.01E+00	3.64E−08
F_{17}	1.15E−07	3.98E−01	7.61E−04	3.98E−01	1.13E−16	3.98E−01	**3.71E−17**	3.98E−01	9.45E−15
F_{18}	1.48E+01	3.00E+00	2.25E−05	3.00E+00	3.24E−02	3.00E+00	6.33E−07	3.00E+00	**1.94E−10**
F_{19}	3.53E−07	−3.75E+00	2.55E−03	−3.86E+00	4.15E−01	−3.81E+00	4.37E−10	−3.73E+00	9.69E−04
F_{20}	**5.37E−02**	−2.84E+00	3.71E−01	−1.47E+00	5.32E−01	−2.39E+00	4.37E−01	−2.17E+00	1.64E−01
F_{21}	2.91E+00	−2.28E+00	1.80E+00	−4.57E+00	1.30E+00	−5.19E+00	2.34E+00	−7.33E+00	1.29E+00
F_{22}	3.02E+00	−3.99E+00	1.99E+00	−6.58E+00	2.64E+00	−2.97E+00	1.37E−02	−1.00E+00	2.89E−04
F_{23}	3.13E+00	−4.49E+00	1.96E+00	−9.37E+00	2.75E+00	−3.10E+00	2.37E+00	−2.46E+00	1.19E+00

Table 10
Results of composite benchmark test functions.

F	EPO		SHO		GWO		PSO		MVO		SCA		GSA		GA		HS	
	Ave	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*
F_{24}	2.33E+02	9.22E+01	2.30E+02	1.37E+02	8.39E+01	8.42E+01	6.00E+01	8.94E+01	1.40E+02	1.52E+02	1.20E+02	3.11E+01	**4.49E−17**	**2.56E−17**	5.97E+02	1.34E+02	4.37E+01	2.09E+01
F_{25}	**4.10E+01**	3.09E+01	4.08E+02	9.36E+01	1.48E+02	3.78E+01	2.44E+02	1.73E+02	2.50E+02	1.44E+02	1.14E+02	**1.84E+00**	2.03E+02	4.47E+02	4.09E+02	2.10E+01	1.30E+02	3.34E+00
F_{26}	**3.39E+02**	5.03E+01	3.39E+02	**3.14E+01**	3.53E+02	5.88E+01	3.39E+02	8.36E+01	4.05E+02	1.67E+02	3.89E+02	5.41E+01	3.67E+02	8.38E+01	9.30E+02	8.31E+01	5.07E+02	4.70E+01
F_{27}	**3.44E+02**	**2.17E+01**	7.26E+02	1.21E+02	4.23E+02	1.14E+02	4.49E+02	1.42E+02	3.77E+02	1.28E+02	4.31E+02	2.94E+01	5.32E+02	1.01E+02	4.97E+02	3.24E+01	3.08E+02	2.07E+02
F_{28}	1.46E+02	7.06E+01	**1.06E+02**	**1.38E+01**	1.36E+02	2.13E+02	2.40E+02	4.25E+02	2.45E+02	9.96E+01	1.56E+02	8.30E+01	1.44E+02	1.31E+02	1.90E+02	5.03E+01	8.43E+02	6.94E+02
F_{29}	7.43E+02	9.09E+02	**5.97E+02**	**4.98E+00**	8.26E+02	1.74E+02	8.22E+02	1.80E+02	8.33E+02	1.68E+02	6.06E+02	1.66E+02	8.13E+02	1.13E+02	6.65E+02	3.37E+02	7.37E+02	2.76E+01

3.4.2. Space complexity

The space complexity of EPO algorithm is the maximum amount of space used at any one time which is considered during its initialization process. Thus, the total space complexity of EPO algorithm is $\mathcal{O}(n \times d)$.

4. Experimental results and discussion

This section describes the experimentation on forty-four standard benchmark test functions to evaluate the performance of proposed algorithm. The detailed description of these benchmarks are presented below. Further, the results are compared with eight well-known metaheuristic algorithms.

4.1. Benchmark test functions

The forty-four benchmark test functions are applied on proposed algorithm to demonstrate its applicability and efficiency. These functions are divided into five main categories: Unimodal [51], Multimodal [33], Fixed-dimension Multimodal [33,51], Composite functions [52], and CEC 2015 functions [53]. These functions are tabulated in Appendix A. In Appendix A, *Dim* and *Range* indicate the dimension of the function and boundary of the search space, respectively whereas $f_{\min}$ denotes the minimization function.

Appendices A.1–A.3 show the characteristics of unimodal, multimodal, and fixed-dimension multimodal benchmark test functions, respectively. The detailed description of composite benchmark test functions is tabulated in Table 10. Whereas, the detailed description of CEC 2015 benchmark test functions is presented in Table 11. The seven test functions ($F_1 - F_7$) are included in the category of unimodal test functions. These functions have only one global optimum.

The second category consists of six test functions ($F_8 - F_{13}$) and third category includes ten test functions ($F_{14} - F_{23}$). There are multiple local solutions in these categories which are useful for examining the local optima problem.

The fourth category comprises six composite test functions ($F_{24} - F_{29}$) which are the shifted, rotated, expanded, and combined version of classical functions [54].

The fifth category consists of fifteen CEC 2015 test functions ($CEC1 - CEC15$) which are shifted, rotated, hybrid, and composition functions as described in Appendix A.

4.2. State-of-the-art algorithms for comparison

To validate the performance of the proposed EPO algorithm, the eight well-known optimization algorithms are used for comparison.

- *Spotted Hyena Optimizer (SHO)* [29,55,56]: Spotted Hyena Optimizer (SHO) is a recently developed bio-inspired based optimization algorithm proposed by Dhiman and Kumar [29]. It mimics the searching, encircling, and hunting behaviors of spotted hyena. The main concept of this algorithm is the social relationship between the spotted hyenas. In SHO, the search agents can update their positions with a set of solutions (i.e., group or cluster of optimal solutions) rather than one optimal solution. This algorithm was applied on constrained as well as unconstrained real-life engineering problems and well-known test functions.
- *Grey Wolf Optimizer (GWO)* [57]: Grey Wolf Optimizer (GWO) is a very popular bio-inspired based algorithm for solving real-life constrained problems. Grey Wolf Optimizer (GWO) is inspired by the behaviors of grey wolves. It mimics the leadership, hierarchy, and hunting mechanisms of grey wolves. GWO employed four types of grey wolves: alpha, beta, delta, and omega for optimization problems. The hunting, searching, encircling, and attacking mechanisms are also implemented. Further, to investigate the performance of GWO algorithm, it was tested on well-known test functions and classical engineering design problems.

- *Particle Swarm Optimization (PSO)* [24]: Particle Swarm Optimization (PSO) is also another population based stochastic optimization algorithm which is inspired by the social behavior of fish schooling or bird flocking. Each particle can move through out the search space with respect to global best solution and updates its current position if it is better than previous best solution. The reason behind the popularity of this algorithm is that there are only few parameters to adjust.
- *Multi-Verse Optimizer (MVO)* [58]: Multi-verse Optimizer (MVO) is a promising optimization algorithm proposed by Mirjalili et al. [58]. It is inspired by the theory of multi-verse in physics which consists of three main concepts i.e., white hole, black hole, and worm hole. The concepts of white hole and black hole are appropriate for exploration and worm hole helps for exploitation the search spaces.
- *Sine Cosine Algorithm (SCA)* [59]: Sine Cosine Algorithm (SCA) is proposed by Mirjalili for solving numerical optimization problems. The SCA generates multiple random solutions and fluctuate them towards the best optimal solution using mathematical models such as sine and cosine functions. The convergence speed of SCA is very high which is helpful for local optima avoidance.
- *Gravitational Search Algorithm (GSA)* [9]: Gravitational Search Algorithm (GSA) is proposed by Rashedi et al. [9] which is based on the Newton's law of gravitation and law of motion. In GSA, the agent has four parameters: position, inertial mass, active gravitational mass, and passive gravitational mass. Last few years, researchers have applied the GSA algorithm on large scale problems because it has an ability to find global optimum and requires only two parameters which is far less as compared to other nature-inspired algorithms.
- *Genetic Algorithm (GA)* [60]: Genetic Algorithm (GA) is a evolutionary algorithm inspired by the theory of natural selection. GA evolves three operators such as selection, crossover, and mutation which are frequently used to find the near optimal solutions. The algorithm evolves better solutions over generations until the stopping criterion is satisfied.
- *Harmony Search (HS)* [61]: The Harmony Search (HS) algorithm was originally inspired by the process of Jazz musicians. HS is a population based algorithm that maintains a set of solutions in the Harmony Memory (HM). It imposes only few mathematical requirements and less initial value settings of the decision variables. HS is an efficient for finding a set of solutions in a large search region.

4.3. Experimental setup

The parameter settings of proposed EPO and competitor metaheuristic algorithms i.e., SHO, GWO, PSO, MVO, SCA, GSA, GA, and HS are tabulated in the Table 1. The parameter values of these algorithms are set as they are recommended in their original papers. The experimentation has been done on Matlab R2014a (8.3.0.532) version in the environment of Microsoft Windows 8.1 using 64 bit Core i-5 processor with 2.40 GHz and 4 GB main memory. The average and standard deviation of the best optimal solution are mentioned in tables. For each benchmark test function, EPO algorithm utilizes 30 independent runs in which each run employs 1000 iterations.

4.4. Performance comparison

In order to demonstrate the effectiveness of proposed algorithm, it is compared with eight well-known optimization algorithms on unimodal, multimodal, fixed-dimension multimodal, composite, and CEC 2015 benchmark test functions.

4.4.1. Evaluation of test functions $F_1 - F_7$

The functions $F1 - F7$ are unimodal test problems that are used to assess the exploitation capability of the metaheuristic algorithms. Table 7 shows that EPO is very competitive as compared with other

Table 11
Results of CEC 2015 benchmark test functions.

F	EPO		SHO		GWO		PSO		MVO		SCA		GSA		GA		HS	
	Ave	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*
CEC − 1	**1.50E+05**	1.21E+06	2.28E+06	2.18E+06	2.02E+06	2.08E+06	4.37E+05	**4.73E+05**	1.47E+06	2.63E+06	6.06E+05	5.02E+05	7.65E+06	3.07E+06	3.20E+07	8.37E+06	8.89E+06	6.95E+06
CEC − 2	6.70E+06	1.34E+08	3.13E+05	4.19E+05	5.65E+06	6.03E+06	9.41E+03	1.08E+04	1.97E+04	1.46E+04	1.43E+04	1.03E+04	7.33E+08	2.33E+08	**4.58E+03**	**1.09E+03**	2.97E+05	2.85E+03
CEC − 3	**3.20E+02**	1.16E−03	3.20E+02	3.76E−02	3.20E+02	7.08E−02	3.20E+02	8.61E−02	3.20E+02	9.14E−02	3.20E+02	3.19E−02	3.20E+02	7.53E−02	3.20E+02	**1.11E−05**	3.20E+02	2.78E−02
CEC − 4	4.10E+02	5.61E+01	4.11E+02	1.71E+01	4.16E+02	1.03E+01	**4.09E+02**	**3.96E+00**	4.26E+02	1.17E+01	4.18E+02	1.03E+01	4.42E+02	7.72E+00	4.39E+02	7.25E+00	6.99E+02	6.43E+00
CEC − 5	9.81E+02	2.06E+02	9.13E+02	1.85E+02	9.20E+02	**1.78E+02**	**8.65E+02**	2.16E+02	1.33E+03	3.45E+02	1.09E+03	2.81E+02	1.76E+03	2.30E+02	1.75E+03	2.79E+02	1.26E+03	1.86E+02
CEC − 6	2.05E+03	1.05E+04	1.29E+04	1.15E+04	2.26E+04	2.45E+04	**1.86E+03**	**1.93E+03**	7.35E+03	3.82E+03	3.82E+03	2.44E+03	2.30E+04	2.41E+04	3.91E+06	2.70E+06	2.91E+05	1.67E+05
CEC − 7	**7.02E+02**	**5.50E−01**	7.02E+02	6.76E−01	7.02E+02	7.07E−01	7.02E+02	7.75E−01	7.02E+02	1.10E+00	7.02E+02	9.40E−01	7.06E+02	9.07E−01	7.08E+02	1.32E+00	7.08E+02	2.97E+00
CEC − 8	**1.47E+03**	2.34E+03	1.86E+03	1.98E+03	3.49E+03	2.04E+03	3.43E+03	2.77E+03	9.93E+03	8.74E+03	2.58E+03	**1.61E+03**	6.73E+03	3.36E+03	6.07E+05	4.81E+05	5.79E+04	2.76E+04
CEC − 9	**1.00E+03**	1.51E+01	1.00E+03	1.43E−01	1.00E+03	1.28E−01	1.00E+03	7.23E−02	1.00E+03	2.20E−01	1.00E+03	**5.29E−02**	1.00E+03	9.79E−01	1.00E+03	5.33E+00	1.00E+03	3.97E+00
CEC − 10	**1.23E+03**	2.51E+04	2.00E+03	2.73E+03	4.00E+03	2.82E+03	3.27E+03	1.84E+03	8.39E+03	1.12E+04	2.62E+03	**1.78E+03**	9.91E+03	8.83E+03	3.42E+05	1.74E+05	4.13E+04	2.39E+04
CEC − 11	**1.35E+03**	**1.41E+01**	1.38E+03	2.42E+01	1.40E+03	5.81E+01	1.35E+03	1.12E+02	1.37E+03	8.97E+01	1.39E+03	5.42E+01	1.35E+03	1.11E+02	1.41E+03	7.73E+01	1.36E+03	5.39E+01
CEC − 12	**1.30E+03**	7.50E+00	1.30E+03	7.89E−01	1.30E+03	**6.69E−01**	1.30E+03	6.94E−01	1.30E+03	9.14E−01	1.30E+03	8.07E−01	1.31E+03	1.54E+00	1.31E+03	2.05E+00	1.31E+03	1.65E+00
CEC − 13	**1.30E+03**	**6.43E−05**	1.30E+03	2.76E−04	1.30E+03	1.92E−04	1.30E+03	5.44E−03	1.30E+03	1.04E−03	1.30E+03	2.43E−04	1.30E+03	3.78E−03	1.35E+03	4.70E+01	1.35E+03	3.97E+01
CEC − 14	**3.22E+03**	2.12E+03	4.25E+03	1.73E+03	7.29E+03	2.45E+03	7.10E+03	3.12E+03	7.60E+03	1.29E+03	7.34E+03	2.47E+03	7.51E+03	1.52E+03	9.30E+03	**4.04E+02**	8.96E+03	6.32E+03
CEC − 15	**1.60E+03**	5.69E+01	1.60E+03	3.76E+00	1.61E+03	4.94E+00	1.60E+03	**2.66E−07**	1.61E+03	1.13E+01	1.60E+03	1.80E−02	1.62E+03	3.64E+00	1.64E+03	1.12E+01	1.63E+03	3.67E+01

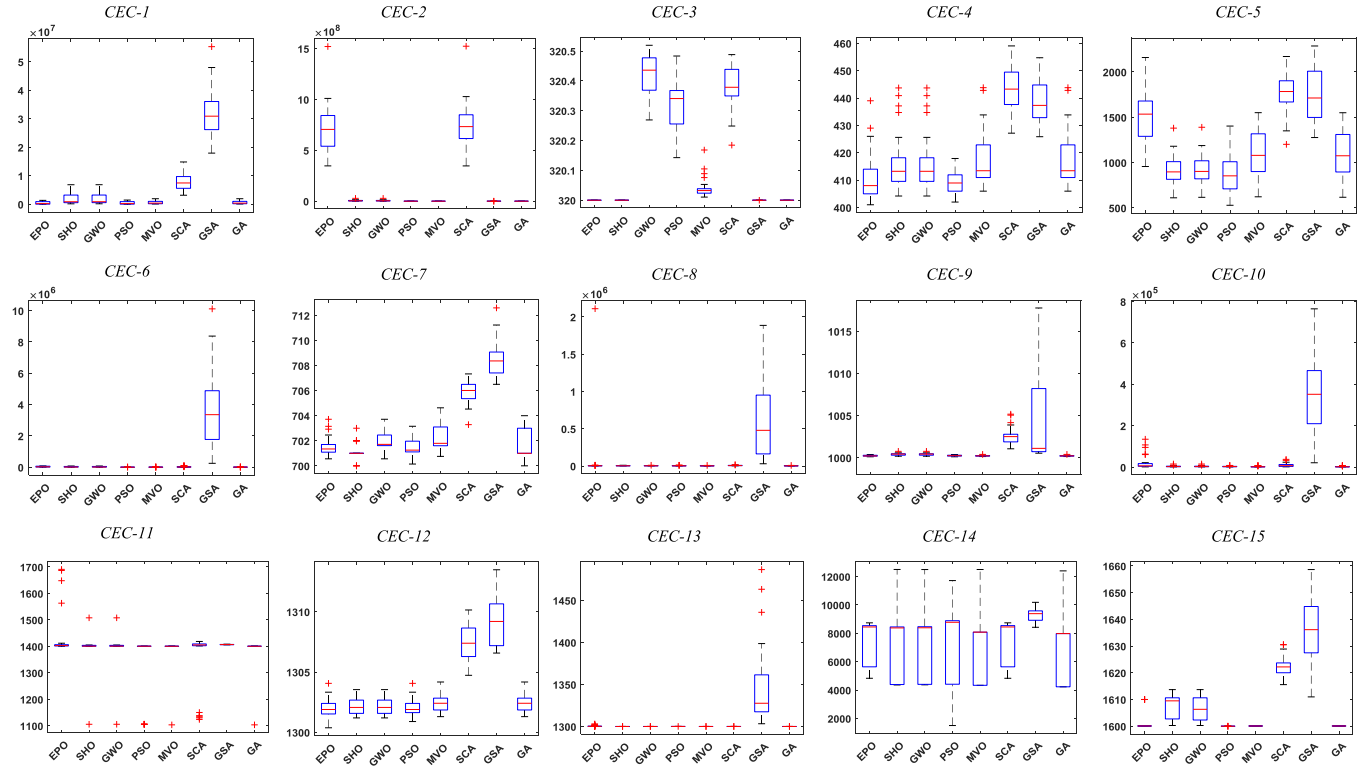

Fig. 11. Boxplot obtained from proposed EPO and other competitor algorithms on CEC 2015 benchmark test functions.

competitor algorithms. The results reveal that EPO is able to determine the optimal solution for functions F_4, F_5, F_6, and F_7. EPO has better exploitation capability and able to find the best optimal solution very efficiently.

4.4.2. Evaluation of test functions $F_8 - F_{23}$

Multimodal test functions have an ability to evaluate the exploration of an optimization algorithm. Tables 8 and 9 depict the performance of above-mentioned algorithms on multimodal test functions ($F_8 - F_{13}$) and fixed-dimension multimodal test functions ($F_{14} - F_{23}$). From these tables, it can be seen that EPO is able to find optimal solution for nine test problems (i.e., F_8, F_{10}, F_{13}, F_{14}, F_{15}, F_{17}, F_{18}, F_{19}, and F_{22}) and also obtains competitive results in majority of test problems. The results reveal that EPO algorithm has better exploration capability.

4.4.3. Evaluation of test functions $F_{24} - F_{29}$

The composite benchmark functions is very challenging task because these functions require balance between exploration and exploitation. These test functions are shifted, rotated, expanded, and combined version of classical functions. The local optima avoidance can be observed in these test functions. Table 10 depicts that EPO algorithm is an effective optimizer for F_{25}, F_{26}, and F_{27} test functions and very competitive in other test cases.

4.4.4. Evaluation of CEC 2015 test functions (CEC1 - CEC15)

This special session is devoted to the real approaches and techniques for solving single objective optimization problems. All of these test functions are minimization problems which should be treated as black-box problems with bound constraints. Table 11 shows the performance of EPO and other algorithms on CEC 2015 test functions. It is observed from Table 11 that proposed algorithm is efficient for $CEC-1$, $CEC-3$, $CEC-7$, $CEC-8$, $CEC-9$, $CEC-10$, $CEC-11$, $CEC-12$, $CEC-13$, $CEC-14$, and $CEC-15$ test functions. Fig. 11 shows the box plot for the proposed EPO and other competitor algorithms on CEC 2015 benchmark test functions. It can be seen from this figure that EPO is the best efficient optimizer for most of the benchmark test functions.

4.5. Convergence analysis

The main intention behind the convergence analysis is to understand the behavior of proposed EPO algorithm. The search agents explore the whole search space and changes rapidly in the initial stage of optimization process.

Fig. 12 shows the convergence curves of EPO and other metaheuristic algorithms. It is shown that EPO is very competitive over other metaheuristic techniques. EPO has three different convergence behaviors.

In the initial stage of iterations, EPO converges more quickly in the search space due to its adaptive mechanism. This behavior is shown in test functions (i.e., $F1$, $F9$, $F12$, and $F18$).

In second step, EPO converges towards the optimum when final iteration reaches which is shown in $F23$ and $F27$ test functions.

The last step shows the express convergence from the initial step of iterations which is shown in $F5$ and $F15$ test functions. These results reveal that EPO algorithm maintains a proper balance between exploration and exploitation to find the global optimum.

Hence, the results disclose different behaviors of EPO algorithm which show that the success rate of EPO is better than the other competitor algorithms for solving optimization problems.

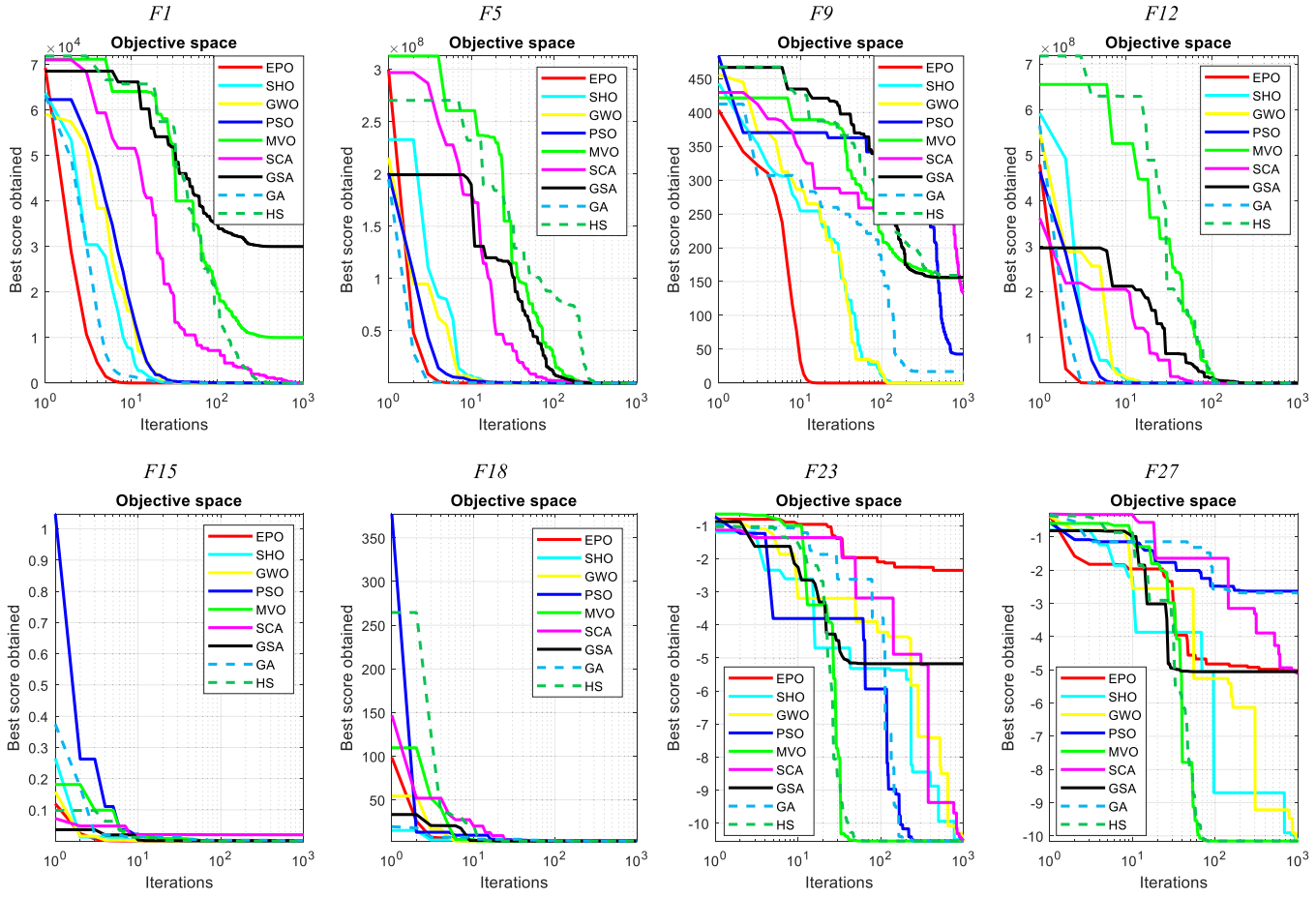

Fig. 12. Convergence analysis between the proposed EPO and other competitive algorithms on benchmark test problems.

4.6. Sensitivity analysis

The proposed EPO algorithm involves four parameters: maximum number of iterations, number of search agents, parameter f, and parameter l. The sensitivity investigation of these parameters has been discussed by varying their values and keeping other parameters fixed as mentioned in Section 4.3 and Table 1.

1. Maximum number of iterations: EPO algorithm was run for different number of iterations. The values of $Max_{iteration}$ used in experimentation are 100, 500, 800, and 1000. Table 2 and Fig. 10(a) show the effect of number of iterations over benchmark test functions. The results show that EPO converges towards the optimum when the number of iterations is increased.
2. Number of search agents: To investigate the effect of number of search agents on benchmark test functions, EPO algorithm was executed for 30, 50, 80, and 100. Table 3 and Fig. 10(b) show the effect of number of search agents on benchmark test functions. The results show that EPO provides best optimal solutions when the number of search agent is set to 80.
3. Variation in parameter M: The proposed EPO was executed for different values of M keeping other parameters fixed i.e., maximum number of iterations and number of search agents. The values of parameter M are varied from [1, 2, 3, 4]. Table 4 and Fig. 10(c) show the effect of M over different benchmark test functions. The results demonstrate that EPO achieves near optimal solutions when the value of M is set to 2.
4. Variation in parameter f: EPO algorithm was run for different values of f keeping other parameters fixed i.e., maximum number of iterations and number of search agents. The values of f used in experimentation are lies in the ranges of [1, 2], [2, 3], [3, 4], and [4, 5]. Table 5 and Fig. 10(d) show the effect of f over benchmark test functions. The results reveal that EPO provides better results when the value of f is set between the range of [2, 3].
5. Variation in parameter l: The values used in experimentation for l are lie between the ranges of [0.5, 1], [1, 1.5], [1.5, 2], and [2, 2.5]. Table 6 and Fig. 10(e) show the effect of l on different benchmark test functions. It is been observed that EPO provides optimal results when the value of l is fixed between the range of [1.5, 2].

4.7. Scalability study

This section describes the effect of scalability on various test functions using proposed EPO. The dimensionality of the test functions varies from 30 to 50, 50 to 80, and 80 to 100. Fig. 13 shows the performance of EPO algorithm on scalable benchmark test functions. It is observed that the performance of EPO is not too degraded when the dimensionality of search space is increased. The results reveal that the performance of EPO is least affected with the increase in dimensionality of search space.

4.8. Statistical testing

Apart from standard statistical analysis such as mean and standard deviation, ANOVA test has been conducted. The ANOVA test is used to determine whether the results obtained from proposed algorithm are different from other competitor algorithms in a statistically significant way. The sample size for ANOVA test is 30 with 95% confidence of interval. The results of ANOVA test are tabulated in Table 12 which shows that EPO is statistically significant from other algorithms for all

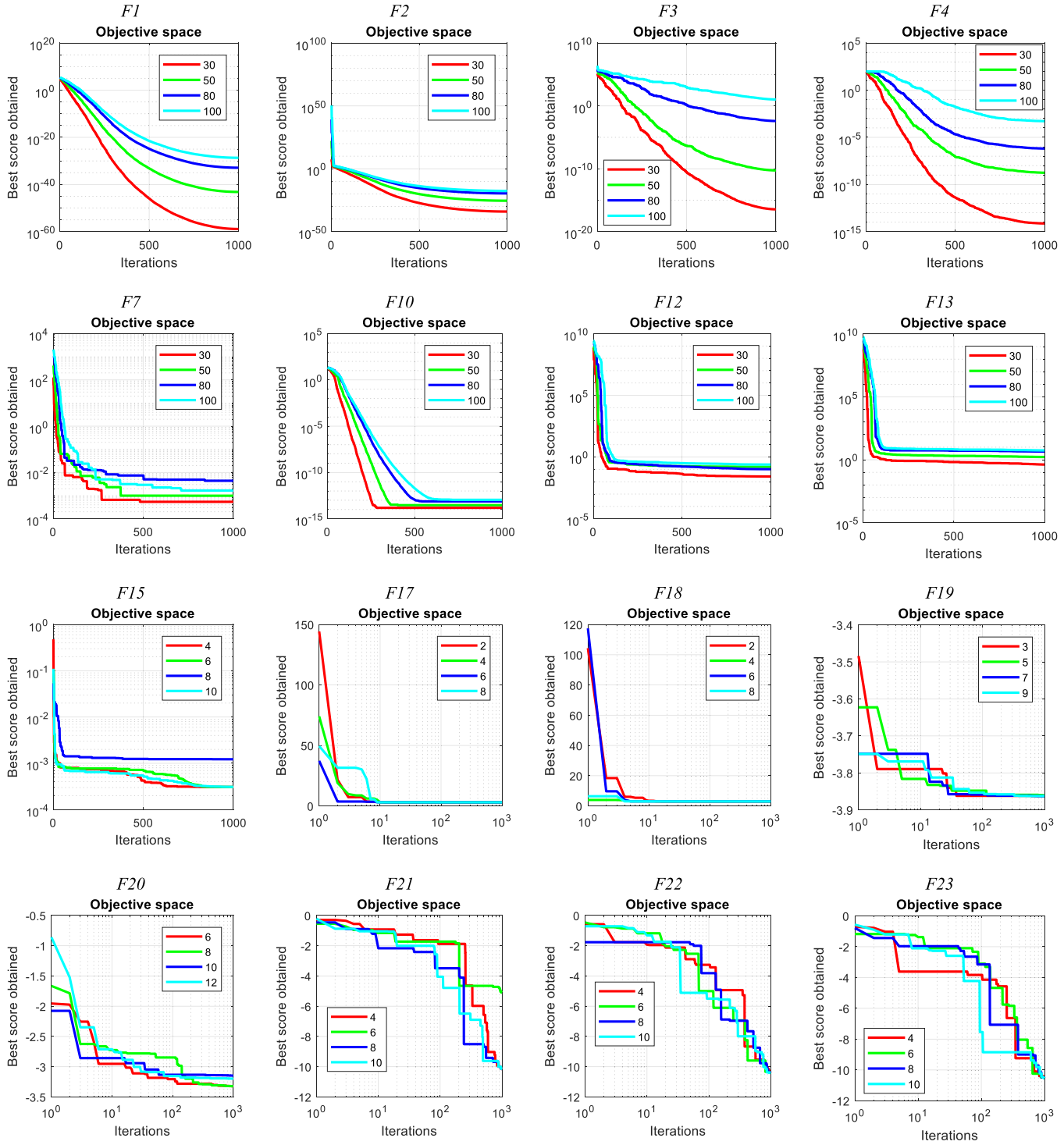

Fig. 13. Effect of scalability on the performance of EPO algorithm.

benchmark test functions.

5. EPO For engineering design problems

In this section, the proposed EPO algorithm has been applied on six constrained and one unconstrained non-linear engineering design problems. These are pressure vessel, speed reducer, welded beam, tension/compression spring, 25-bar truss, rolling element bearing, and displacement of loaded structure. All of these optimization problems have different constraints with different nature. There are different types of penalty functions to handle these problems [62]: Static penalty, Dynamic penalty, Annealing penalty, Adaptive penalty, Co-evolutionary penalty, and Death penalty. However, death penalty function handles the solution which can violate the constraints and assigns zero fitness value to discard the infeasible solutions during optimization. Therefore, this function does not employ any information of infeasible solutions. Due to its simplicity and low computational complexity, EPO algorithm is equipped with death penalty function to handle both constrained and unconstrained design problems.

Table 12
ANOVA test results.

F	p−value	EPO	SHO	GWO	PSO	MVO	SCA	GSA	GA	HS
F_1	2.72E−24	SHO, GWO, PSO, MVO, SCA, GSA, GA, HS	EPO, GWO, PSO, MVO, SCA, GSA, GA, HS	EPO, SHO, PSO, MVO, SCA, GSA, GA, HS	EPO, SHO, GWO, MVO, SCA, GSA, GA, HS	EPO, SHO, GWO, PSO, SCA, GSA, GA, HS	EPO, SHO, GWO, PSO, MVO, GSA, GA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GSA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GSA, GA
F_2	3.60E−21	SHO, GWO, PSO, MVO, SCA, GA, HS	EPO, GWO, PSO, MVO, SCA, GSA, GA, HS	EPO, SHO, PSO, MVO, SCA, GSA, GA, HS	EPO, SHO, GWO, MVO, SCA, GSA, GA, HS	EPO, SHO, GWO, PSO, SCA, GSA, GA, HS	EPO, SHO, PSO, MVO, GSA, GA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GA, HS	EPO, SHO, GWO, PSO, SCA, GSA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GSA
F_3	5.53E−30	SHO, GWO, PSO, SCA, GSA, GA, HS	EPO, GWO, PSO, SCA, GSA, GA	EPO, SHO, PSO, MVO, SCA, GSA, GA, HS	EPO, SHO, GWO, MVO, SCA, GSA, GA, HS	EPO, GWO, PSO, SCA, GSA, GA, HS	EPO, SHO, GWO, MVO, GSA, GA, HS	EPO, SHO, GWO, PSO, MVO, SCA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GSA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GA
F_4	1.80E−74	SHO, GWO, PSO, MVO, SCA, GSA, GA, HS	EPO, GWO, PSO, MVO, SCA, GSA, GA, HS	EPO, SHO, PSO, MVO, SCA, GSA, GA, HS	EPO, SHO, GWO, MVO, SCA, GSA, GA, HS	EPO, SHO, GWO, PSO, SCA, GSA, GA, HS	EPO, SHO, GWO, PSO, MVO, GSA, GA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GSA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GSA, GA
F_5	2.11E−11	SHO, GWO, PSO, MVO, SCA, GSA, GA, HS	EPO, PSO, MVO, SCA, GSA, GA, HS	EPO, SHO, PSO, MVO, SCA, GSA, GA, HS	EPO, SHO, GWO, MVO, SCA, GSA, GA, HS	EPO, SHO, GWO, PSO, SCA, GSA, GA, HS	EPO, SHO, GWO, PSO, GSA, GA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GA, HS	EPO, GWO, PSO, MVO, SCA, GSA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GSA, GA
F_6	4.97E−19	SHO, GWO, PSO, MVO, SCA, GSA, GA, HS	EPO, GWO, PSO, MVO, SCA, GA, HS	EPO, SHO, PSO, MVO, SCA, GSA, GA, HS	EPO, SHO, MVO, SCA, GSA, GA, HS	EPO, SHO, GWO, PSO, SCA, GSA, GA, HS	EPO, SHO, GWO, MVO, GSA, GA, HS	EPO, SHO, GWO, PSO, SCA, GA, HS	EPO, SHO, GWO, PSO, SCA, GSA, HS	EPO, SHO, GWO, PSO, MVO, GSA, GA
F_7	1.13E−85	SHO, GWO, PSO, MVO, SCA, GSA, HS	EPO, GWO, PSO, MVO, SCA, GSA, GA, HS	EPO, SHO, PSO, MVO, SCA, GA, HS	EPO, SHO, MVO, SCA, GSA, GA, HS	EPO, SHO, GWO, PSO, SCA, GSA, GA, HS	EPO, SHO, GWO, MVO, GSA, GA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GSA	EPO, SHO, GWO, PSO, SCA, GSA, GA
F_8	7.56E−14	SHO, GWO, PSO, MVO, SCA, GA, HS	EPO, GWO, PSO, MVO, SCA, GSA, HS	EPO, SHO, PSO, MVO, SCA, GA, HS	EPO, SHO, MVO, SCA, GSA, GA	EPO, SHO, GWO, PSO, SCA, GSA, GA, HS	EPO, SHO, GWO, PSO, GSA, HS	EPO, SHO, PSO, MVO, SCA, GA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GSA, HS	EPO, SHO, PSO, MVO, SCA
F_9	1.00E−95	SHO, GWO, PSO, MVO, SCA, GSA, GA, HS	EPO, GWO, PSO, MVO, GSA, GA, HS	EPO, SHO, PSO, MVO, SCA, GSA, GA, HS	EPO, SHO, SCA, GSA, GA, HS	EPO, SHO, PSO, SCA, GSA, GA, HS	EPO, SHO, GWO, PSO, MVO, GSA, GA, HS	EPO, SHO, GWO, PSO, MVO, GA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GSA, HS	EPO, SHO, GWO, PSO, SCA, GSA, GA
F_{10}	1.21E−81	SHO, GWO, PSO, MVO, SCA, GSA, GA, HS	EPO, GWO, PSO, MVO, SCA, GSA, GA, HS	EPO, SHO, PSO, MVO, SCA, GSA, GA, HS	EPO, SHO, GWO, MVO, GSA	EPO, GWO, PSO, SCA, GSA, GA, HS	EPO, SHO, GWO, PSO, MVO, GSA, GA, HS	EPO, GWO, PSO, MVO, SCA, GA, HS	EPO, SHO, GWO, PSO, SCA	EPO, SHO, GWO, PSO, MVO, SCA, GSA, GA
F_{11}	9.36E−07	SHO, GWO, PSO, MVO, SCA, GSA, HS	EPO, GWO, PSO, MVO, SCA, GSA, GA, HS	EPO, SHO, MVO, SCA, GSA, GA, HS	EPO, SHO, GWO, MVO, SCA, GSA, GA, HS	EPO, SHO, GWO, PSO, GSA, GA, HS	EPO, SHO, GWO, PSO, MVO, GSA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GA, HS	EPO, SHO, GWO, PSO, MVO, SCA, HS	EPO, SHO, GWO, PSO, MVO, GSA, GA
F_{12}	2.99E−53	SHO, GWO, PSO, MVO, SCA, GSA, GA, HS	EPO, GWO, PSO, MVO, SCA, GSA, GA, HS	EPO, SHO, MVO, SCA, GSA, GA, HS	EPO, SHO, GWO, MVO, GSA, GA, HS	EPO, SHO, GWO, PSO, SCA, GSA, GA, HS	EPO, SHO, GWO, PSO, GSA, GA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GA, HS	EPO, SHO, PSO, MVO, SCA, GSA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GA
F_{13}	4.90E−38	SHO, GWO, PSO, MVO, SCA, GSA, GA, HS	EPO, GWO, PSO, MVO, SCA, GA, HS	EPO, SHO, PSO, MVO, SCA, GSA, GA, HS	EPO, SHO, GWO, SCA, GSA, GA, HS	EPO, SHO, PSO, SCA, GSA, GA, HS	EPO, SHO, GWO, PSO, MVO, GSA, GA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GA, HS	EPO, SHO, GWO, PSO, MVO, GSA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GA
F_{14}	4.34E−40	SHO, GWO, PSO, MVO, SCA, GSA, HS	EPO, GWO, PSO, MVO, SCA, GSA, GA, HS	EPO, SHO, PSO, MVO, SCA, GSA	EPO, SHO, MVO, SCA, GSA, GA, HS	EPO, SHO, GWO, PSO, SCA, GSA, GA, HS	EPO, SHO, GWO, PSO, GSA, GA, HS	EPO, GWO, PSO, MVO, SCA, GA, HS	EPO, SHO, GWO, PSO, MVO, GSA, HS	EPO, SHO, GWO, PSO, MVO, SCA
F_{15}	3.15E−37	SHO, GWO, PSO, MVO, SCA, GSA, GA, HS	EPO, GWO, PSO, MVO, SCA, GSA, HS	EPO, SHO, PSO, SCA, GSA, GA, HS	EPO, SHO, GWO, MVO, SCA, GSA, GA, HS	EPO, SHO, GWO, PSO, SCA, GSA, GA	EPO, SHO, GWO, PSO, MVO, GA, HS	EPO, SHO, PSO, MVO, SCA, GA, HS	EPO, SHO, GWO, MVO, SCA, GSA	EPO, SHO, GWO, PSO, MVO, SCA
F_{16}	2.07E−10	SHO, GWO, PSO, MVO, SCA, GSA, GA, HS	EPO, GWO, MVO, SCA, GSA, GA, HS	EPO, SHO, PSO, MVO, SCA, GSA, GA, HS	EPO, SHO, GWO, MVO, SCA, GSA, GA, HS	EPO, GWO, PSO, SCA, GSA, GA, HS	EPO, SHO, GWO, PSO, MVO, GSA	EPO, SHO, GWO, PSO, MVO, SCA	EPO, SHO, GWO, PSO, MVO, SCA, GSA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GA
F_{17}	1.70E−76	SHO, GWO, PSO, MVO, SCA, GSA, GA, HS	EPO, GWO, PSO, MVO, SCA, GSA, GA, HS	EPO, SHO, PSO, MVO, GSA, GA, HS	EPO, SHO, GWO, MVO, SCA, GSA, GA, HS	EPO, SHO, GWO, PSO, SCA, GA, HS	EPO, SHO, PSO, MVO, GSA, GA, HS	EPO, SHO, GWO, MVO, SCA, GA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GSA, HS	EPO, SHO, GWO, MVO, SCA, GSA, GA
F_{18}	4.69E−32	SHO, GWO, PSO, MVO, GSA, GA, HS	EPO, GWO, PSO, MVO, SCA, GSA, GA, HS	EPO, SHO, MVO, SCA, GSA, GA, HS	EPO, SHO, GWO, MVO, GSA, GA, HS	EPO, SHO, PSO, SCA, GSA, HS	EPO, SHO, GWO, PSO, GSA, GA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GA, HS	EPO, SHO, PSO, MVO, SCA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GSA, GA

(*continued on next page*)

Table 12 (*continued*)

F	p−value	EPO	SHO	GWO	PSO	MVO	SCA	GSA	GA	HS
F_{19}	4.46E−39	SHO, GWO, PSO, MVO, SCA, GSA, GA, HS	EPO, GWO, PSO, MVO, SCA, GSA, GA, HS	EPO, SHO, PSO, MVO, SCA, GA, HS	EPO, SHO, GWO, MVO, SCA, GSA, GA, HS	EPO, SHO, PSO, SCA, GA, HS	EPO, SHO, GWO, PSO, MVO, GSA, GA, HS	EPO, SHO, GWO, PSO, MVO, SCA, HS	EPO, SHO, PSO, SCA, GSA, HS	EPO, GWO, PSO, MVO, SCA, GSA, GA
F_{20}	4.90E−109	SHO, GWO, PSO, MVO, SCA, GSA	EPO, GWO, PSO, MVO, SCA, GSA, GA, HS	EPO, PSO, MVO, SCA, GSA,	EPO, SHO, GWO, MVO, SCA, GSA, GA, HS	EPO, SHO, GWO, PSO, SCA, GSA, GA, HS	EPO, SHO, PSO, MVO, GSA, GA	EPO, SHO, GWO, PSO, MVO, SCA, GA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GSA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GSA, GA
F_{21}	2.11E−39	SHO, GWO, PSO, MVO, SCA, GSA, GA, HS	EPO, GWO, PSO, MVO, GSA, GA, HS	EPO, SHO, PSO, MVO, SCA, GSA, GA, HS	EPO, SHO, GWO, MVO, SCA, GA, HS	EPO, SHO, GWO, PSO, SCA, GSA, GA, HS	EPO, SHO, GWO, PSO, MVO, GSA, GA, HS	EPO, SHO, GWO, PSO, SCA, GA	EPO, SHO, GWO, PSO, MVO, SCA, GSA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GSA, GA
F_{22}	6.13E−68	SHO, GWO, PSO, MVO, SCA, GSA, GA, HS	EPO, GWO, PSO, MVO, GSA, GA, HS	EPO, SHO, PSO, MVO, SCA, GSA, GA, HS	EPO, GWO, MVO, SCA, GA, HS	EPO, SHO, GWO, PSO, SCA, GSA, GA, HS	EPO, SHO, PSO, GSA, GA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GA, HS	EPO, GWO, PSO, SCA, GSA, HS	EPO, SHO, PSO, MVO, SCA, GSA, GA
F_{23}	2.19E−39	SHO, GWO, PSO, MVO, SCA, GSA, GA, HS	EPO, GWO, PSO, MVO, SCA, GSA, HS	EPO, SHO, PSO, MVO, SCA, GSA, GA, HS	EPO, SHO, GWO, MVO, SCA, GSA, GA, HS	EPO, SHO, PSO, GSA, GA, HS	EPO, SHO, GWO, PSO, MVO, GSA, GA	EPO, SHO, GWO, MVO, SCA, GA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GSA	EPO, SHO, GWO, PSO, MVO, SCA, GSA, GA
F_{24}	5.00E−09	SHO, GWO, PSO, MVO, SCA, GSA, GA, HS	EPO, GWO, PSO, MVO, SCA, GSA, GA	EPO, SHO, PSO, MVO, GSA, GA, HS	EPO, SHO, GWO, MVO, SCA, GSA, GA, HS	EPO, GWO, PSO, SCA, GSA, GA	EPO, SHO, GWO, PSO, GSA, GA, HS	EPO, SHO, GWO, PSO, MVO, SCA, HS	EPO, SHO, PSO, MVO, SCA, GSA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GSA, GA
F_{25}	9.00E−16	SHO, GWO, PSO, MVO, SCA, GSA, GA, HS	EPO, GWO, PSO, MVO, SCA, GSA, GA	EPO, SHO, PSO, MVO, SCA, GSA, GA, HS	EPO, SHO, GWO, MVO, SCA, GSA, HS	EPO, GWO, SCA, GSA, HS	EPO, SHO, GWO, PSO, MVO, GSA, GA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GA	EPO, SHO, PSO, SCA, GSA, HS	EPO, SHO, PSO, SCA, GSA, GA
F_{26}	7.87E−30	SHO, GWO, PSO, MVO, SCA, GSA, GA, HS	EPO, GWO, PSO, MVO, SCA, GSA, GA, HS	EPO, SHO, PSO, MVO, SCA	EPO, SHO, MVO, SCA, GSA, GA, HS	EPO, SHO, GWO, PSO, SCA, GSA, GA, HS	EPO, SHO, GWO, PSO, MVO, GSA, GA, HS	EPO, SHO, PSO, SCA, GA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GSA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GSA
F_{27}	2.18E−35	SHO, GWO, PSO, MVO, SCA, GSA, GA, HS	EPO, GWO, PSO, MVO, SCA, GA, HS	EPO, SHO, PSO, MVO, SCA, GSA, GA, HS	EPO, SHO, GWO, MVO, GSA, GA, HS	EPO, SHO, GWO, PSO, SCA, GSA, GA, HS	EPO, SHO, GWO, PSO, MVO, GSA, GA, HS	EPO, SHO, GWO, PSO, SCA, GA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GSA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GSA
F_{28}	8.38E−49	SHO, GWO, PSO, MVO, SCA, GSA, HS	EPO, GWO, MVO, SCA, GSA, GA, HS	EPO, SHO, PSO, MVO, SCA, GSA, HS	EPO, SHO, GWO, MVO, SCA, GSA, GA, HS	EPO, SHO, PSO, GSA, GA, HS	EPO, SHO, GWO, PSO, MVO, GSA, GA	EPO, GWO, PSO, MVO, SCA, GA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GSA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GSA, GA
F_{29}	2.12E−12	SHO, GWO, MVO, SCA, GSA, GA, HS	EPO, GWO, PSO, MVO, SCA, GSA, GA, HS	EPO, SHO, MVO, SCA, GSA	EPO, SHO, GWO, MVO, SCA, GSA, GA, HS	EPO, SHO, GWO, PSO, SCA, GSA, GA, HS	EPO, SHO, PSO, MVO, GSA, GA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GA, HS	EPO, SHO, GWO, PSO, MVO, SCA, GSA	EPO, SHO, GWO, PSO, MVO, GSA, GA

Table 13
Comparison results for pressure vessel design problem.

Algorithms	Optimum variables				Optimum cost
	T_s	T_h	R	L	
EPO	0.778099	0.383241	40.315121	200.00000	**5880.0700**
SHO	0.778210	0.384889	40.315040	200.00000	5885.5773
GWO	0.779035	0.384660	40.327793	199.65029	5889.3689
PSO	0.778961	0.384683	40.320913	200.00000	5891.3879
MVO	0.845719	0.418564	43.816270	156.38164	6011.5148
SCA	0.817577	0.417932	41.74939	183.57270	6137.3724
GSA	1.085800	0.949614	49.345231	169.48741	11550.2976
GA	0.752362	0.399540	40.452514	198.00268	5890.3279
HS	1.099523	0.906579	44.456397	179.65887	6550.0230

Table 14
Statistical results obtained from different algorithms for pressure vessel design problem.

Algorithms	Best	Mean	Worst	Std. Dev.	Median
EPO	**5880.0700**	**5884.1401**	**5891.3099**	024.341	**5883.5153**
SHO	5885.5773	5887.4441	5892.3207	**002.893**	5886.2282
GWO	5889.3689	5891.5247	5894.6238	013.910	5890.6497
PSO	5891.3879	6531.5032	7394.5879	534.119	6416.1138
MVO	6011.5148	6477.3050	7250.9170	327.007	6397.4805
SCA	6137.3724	6326.7606	6512.3541	126.609	6318.3179
GSA	11550.2976	23342.2909	**33226.2526**	5790.625	24010.0415
GA	5890.3279	6264.0053	7005.7500	496.128	6112.6899
HS	6550.0230	6643.9870	8005.4397	657.523	7586.0085

5.1. Constrained design problems

This section describes the six constrained real-life engineering design problems and compared it with other algorithms. The statistical analysis of these problems are also done to validate the performance of proposed algorithm.

5.1.1. Pressure vessel design

This problem was first proposed by Kannan and Kramer [63] to minimize the total fabrication cost. Fig. 26 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:

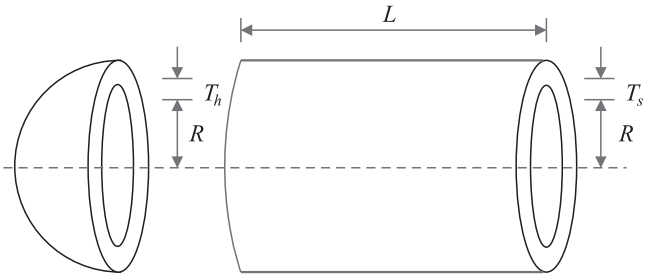

Fig. 14. Schematic view of pressure vessel problem.

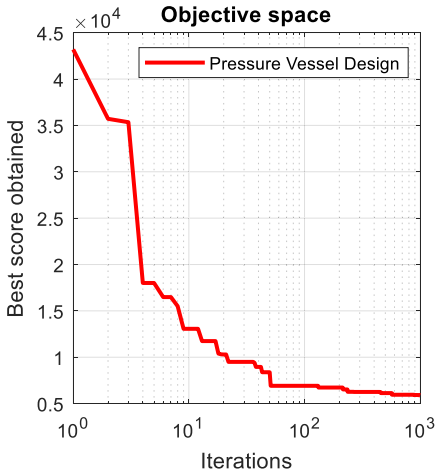

Fig. 15. Convergence analysis of EPO for pressure vessel design problem.

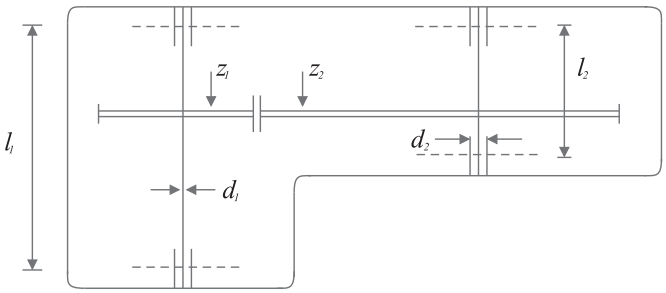

Fig. 16. Schematic view of speed reducer problem.

$$\text{Consider } \vec{z} = [z_1 z_2 z_3 z_4] = [T_s T_h R L],$$
$$\text{Minimize } f(\vec{z}) = 0.6224 z_1 z_3 z_4 + 1.7781 z_2 z_3^2 + 3.1661 z_1^2 z_4 + 19.84 z_1^2 z_3,$$
$$\text{Subject to:}$$
$$g_1(\vec{z}) = -z_1 + 0.0193 z_3 \le 0,$$
$$g_2(\vec{z}) = -z_3 + 0.00954 z_3 \le 0,$$
$$g_3(\vec{z}) = -\pi z_3^2 z_4 - \frac{4}{3}\pi z_3^3 + 1,296,000 \le 0,$$
$$g_4(\vec{z}) = z_4 - 240 \le 0,$$
$$\text{where,}$$
$$1 \times 0.0625 \le z_1, z_2 \le 99 \times 0.0625, 10.0 \le z_3, z_4 \le 200.0. \tag{14}$$

Table 13 shows the comparison of best obtained optimal solution from EPO and other algorithms such as SHO, GWO, PSO, MVO, SCA, GSA, GA, and HS. The proposed EPO provides optimal solution at $z^* = (0.778099, 0.383241, 40.315121, 200.00000)$ with corresponding fitness value equal to $f(z^*) = 5880.0700$. From this table, it can be concluded that, EPO is able to find best optimal design with minimum cost.

The statistical results of pressure vessel design problem are presented in Table 14. The results reveal that EPO performs better than the other competitor algorithms in terms of best mean, and median. Fig. 15 shows the convergence behavior of optimal solution obtained from EPO.

5.1.2. Speed reducer design problem

The speed reducer design problem is a challenging benchmark due to its seven design variables [64] as shown in Fig. 16. The main objective of this problem is to minimize the weight of speed reducer with subject to constraints [65]:

- Bending stress of the gear teeth.
- Surface stress.

Table 15
Comparison results for speed reducer design problem.

Algorithms	Optimum variables							Optimum cost
	b	m	p	l_1	l_2	d_1	d_2	
EPO	3.50123	0.7	17	7.3	7.8	3.33421	5.26536	**2994.2472**
SHO	3.50159	0.7	17	7.3	7.8	3.35127	5.28874	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	5.286715	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	3.461020	5.289213	3030.563
GSA	3.600000	0.7	17	8.3	7.8	3.369658	5.289224	3051.120
GA	3.510253	0.7	17	8.35	7.8	3.362201	5.287723	3067.561
HS	3.520124	0.7	17	8.37	7.8	3.366970	5.288719	3029.002

Table 16
Statistical results obtained from different algorithms for speed reducer design problem.

Algorithms	Best	Mean	Worst	Std. Dev.	Median
EPO	**2994.2472**	**2997.482**	2999.092	**1.78091**	**2996.318**
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	3211.174	79.6381	3105.252
MVO	3002.928	3028.841	3060.958	13.0186	3027.031
SCA	3030.563	3065.917	3104.779	18.0742	3065.609
GSA	3051.120	3170.334	**3363.873**	92.5726	3156.752
GA	3067.561	3186.523	3313.199	17.1186	3198.187
HS	3029.002	3295.329	3619.465	57.0235	3288.657

- 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 comparison of the best obtained optimal solution with various optimization algorithms is given in Table 15. The proposed EPO provides optimal solution at z^*=(3.50123, 0.7, 17, 7.3, 7.8, 3.33421, 5.26536) with corresponding fitness value equal to $f(z^*) = 2994.2472$. However, the statistical results of proposed and eight competitive optimization algorithms including SHO, GWO, PSO, MVO, SCA, GSA, GA, and HS are given in Table 16.

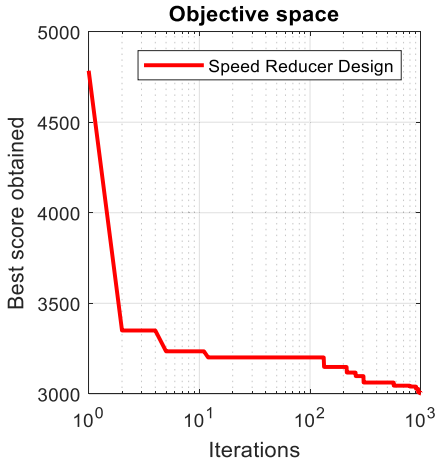

Fig. 17. Convergence analysis of EPO for speed reducer design problem.

The mathematical formulation of this problem is formulated as follows:

$$\text{Consider } \vec{z} = [z_1 z_2 z_3 z_4 z_5 z_6 z_7] = [bmpl_1 l_2 d_1 d_2],$$
$$\text{Minimize } f(\vec{z}) = 0.7854 z_1 z_2^2 (3.3333 z_3^2 + 14.9334 z_3 - 43.0934) - 1.508 z_1 (z_6^2 + z_7^2) + 7.4777 (z_6^3 + z_7^3) + 0.7854 (z_4 z_6^2 + z_5 z_7^2),$$
$$\text{Subject to:}$$
$$g_1(\vec{z}) = \frac{27}{z_1 z_2^2 z_3} - 1 \le 0,$$
$$g_2(\vec{z}) = \frac{397.5}{z_1 z_2^2 z_3^2} - 1 \le 0,$$
$$g_3(\vec{z}) = \frac{1.93 z_4^3}{z_2 z_6^4 z_3} - 1 \le 0,$$
$$g_4(\vec{z}) = \frac{1.93 z_5^3}{z_2 z_7^4 z_3} - 1 \le 0,$$
$$g_5(\vec{z}) = \frac{[(745(z_4/z_2 z_3))^2 + 16.9 \times 10^6]^{1/2}}{110 z_6^3} - 1 \le 0,$$
$$g_6(\vec{z}) = \frac{[(745(z_5/z_2 z_3))^2 + 157.5 \times 10^6]^{1/2}}{85 z_7^3} - 1 \le 0,$$
$$g_7(\vec{z}) = \frac{z_2 z_3}{40} - 1 \le 0,$$
$$g_8(\vec{z}) = \frac{5 z_2}{z_1} - 1 \le 0,$$
$$g_9(\vec{z}) = \frac{z_1}{12 z_2} - 1 \le 0,$$
$$g_{10}(\vec{z}) = \frac{1.5 z_6 + 1.9}{z_4} - 1 \le 0,$$
$$g_{11}(\vec{z}) = \frac{1.1 z_7 + 1.9}{z_5} - 1 \le 0,$$
$$\text{where,}$$
$$2.6 \le z_1 \le 3.6,\ 0.7 \le z_2 \le 0.8,\ 17 \le z_3 \le 28,\ 7.3 \le z_4 \le 8.3,\ 7.3 \le z_5 \le 8.3,\ 2.9 \le z_6 \le 3.9,\ 5.0 \le z_7 \le 5.5. \quad (15)$$

The results indicate that EPO outperforms than other competitor algorithms. Fig. 17 shows that EPO algorithm converges towards the best optimal solution for speed reducer problem.

5.1.3. Welded beam design

The objective of this design problem is to minimize the fabrication cost of welded beam (see Fig. 18). The optimization constraints of welded beam are shear stress (τ) and bending stress (θ) in the beam, buckling load (P_c) on the bar, end deflection (δ) of the beam. There are four design variables of this problem such as

- Thickness of weld (h)
- Length of the clamped bar (l)
- Height of the bar (t)

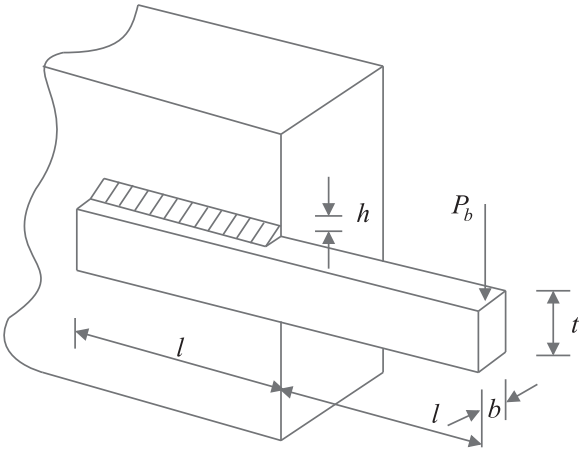

Fig. 18. Schematic view of welded beam problem.

Table 17
Comparison results for welded beam design problem.

Algorithms	Optimum variables				Optimum cost
	h	*l*	*t*	*b*	
EPO	0.205411	3.472341	9.035215	0.201153	**1.723589**
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	0.201395	1.820395
MVO	0.205611	3.472103	9.040931	0.205709	1.725472
SCA	0.204695	3.536291	9.004290	0.210025	1.759173
GSA	0.147098	5.490744	10.00000	0.217725	2.172858
GA	0.164171	4.032541	10.00000	0.223647	1.873971
HS	0.206487	3.635872	10.00000	0.203249	1.836250

- Thickness of the bar (b)

The mathematical formulation is described as follows:

$$\text{Consider } \vec{z} = [z_1 z_2 z_3 z_4] = [hltb],$$
$$\text{Minimize } f(\vec{z}) = 1.10471 z_1^2 z_2 + 0.04811 z_3 z_4 (14.0 + z_2),$$
$$\text{Subject to:}$$
$$g_1(\vec{z}) = \tau(\vec{z}) - 13,600 \le 0,$$
$$g_2(\vec{z}) = \sigma(\vec{z}) - 30,000 \le 0,$$
$$g_3(\vec{z}) = \delta(\vec{z}) - 0.25 \le 0,$$
$$g_4(\vec{z}) = z_1 - z_4 \le 0,$$
$$g_5(\vec{z}) = 6000 - P_c(\vec{z}) \le 0,$$
$$g_6(\vec{z}) = 0.125 - z_1 \le 0,$$
$$g_7(\vec{z}) = 1.10471 z_1^2 + 0.04811 z_3 z_4 (14.0 + z_2) - 5.0 \le 0,$$
$$\text{where,}$$
$$0.1 \le z_1, 0.1 \le z_2, z_3 \le 10.0, z_4 \le 2.0,$$
$$\tau(\vec{z}) = \sqrt{(\tau')^2 + (\tau'')^2 + (l\tau'\tau'')/\sqrt{0.25(l^2 + (h+t)^2)}},$$
$$\tau' = \frac{6,000}{\sqrt{2}hl}, \quad \sigma(\vec{z}) = \frac{504,000}{t^2 b}, \quad \delta(\vec{z}) = \frac{65,856,000}{(30 \times 10^6) b t^3},$$
$$\tau'' = \frac{6,000(14 + 0.5l)\sqrt{0.25(l^2 + (h+t)^2)}}{2[0.707hl(l^2/12 + 0.25(h+t)^2)]},$$
$$P_c(\vec{z}) = 64,746.022(1 - 0.0282346t) t b^3. \quad (16)$$

Table 18
Statistical results obtained from different algorithms for welded beam design problem.

Algorithms	Best	Mean	Worst	Std. Dev.	Median
EPO	**1.723589**	**1.725124**	1.727211	0.004325	**1.724399**
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	**3.048231**	0.324525	2.244663
MVO	1.725472	1.729680	1.741651	0.004866	1.727420
SCA	1.759173	1.817657	1.873408	0.027543	1.820128
GSA	2.172858	2.544239	3.003657	0.255859	2.495114
GA	1.873971	2.119240	2.320125	0.034820	2.097048
HS	1.836250	1.363527	2.035247	0.139485	1.9357485

The comparison for best solution obtained from proposed EPO and other algorithms is presented in Table 17. Among other algorithms, the proposed EPO provides optimal solution at $z^* = (0.205411, 3.472341, 9.035215, 0.201153)$ with corresponding fitness value equal to $f(z^*) = 1.723589$. The results indicate that EPO converges towards the best design. The statistical comparison of the proposed algorithm with other competitor algorithms is depicted in Table 18. It is observed from Table 18 that EPO surpassed other algorithms for providing the best solution in terms of best, mean, and median.

Fig. 19 shows the convergence analysis of best optimal solution obtained from EPO.

5.1.4. Tension/compression spring design problem

The main objective of this problem is to minimize the tension/compression spring weight as shown in Fig. 20. The optimization constraints of this problem are described as follows:

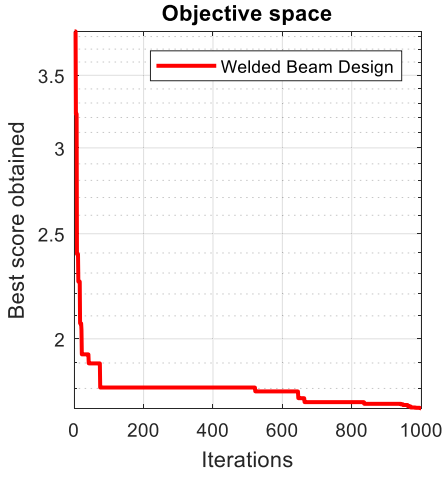

Fig. 19. Convergence analysis of EPO for welded beam design problem.

Fig. 20. Schematic view of tension/compression spring problem.

Table 19
Comparison results for tension/compression spring design problem.

Algorithms	Optimum variables			Optimum cost
	d	*D*	*P*	
EPO	0.051087	0.342908	12.0898	**0.012656987**
SHO	0.051144	0.343751	12.0955	0.012674000
GWO	0.050178	0.341541	12.07349	0.012678321
PSO	0.05000	0.310414	15.0000	0.013192580
MVO	0.05000	0.315956	14.22623	0.012816930
SCA	0.050780	0.334779	12.72269	0.012709667
GSA	0.05000	0.317312	14.22867	0.012873881
GA	0.05010	0.310111	14.0000	0.013036251
HS	0.05025	0.316351	15.23960	0.012776352

- 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:

$$\begin{aligned}&\text{Consider } \vec{z} = [z_1 z_2 z_3] = [dDP],\\ &\text{Minimize } f(\vec{z}) = (z_3 + 2) z_2 z_1^2,\\ &\text{Subject to:}\end{aligned} \tag{17}$$

$$g_1(\vec{z}) = 1 - \frac{z_2^3 z_3}{71785 z_1^4} \le 0,$$

$$g_2(\vec{z}) = \frac{4z_2^2 - z_1 z_2}{12566(z_2 z_1^3 - z_1^4)} + \frac{1}{5108 z_1^2} \le 0,$$

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

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

where,

$0.05 \le z_1 \le 2.0$, $0.25 \le z_2 \le 1.3$, $2.0 \le z_3 \le 15.0$.

The comparison for the best solution obtained from the proposed EPO and other competitor algorithms is presented in Table 19. The best solution was obtained by EPO at design variables $z^* = (0.051087, 0.342908, 12.0898)$ with an objective function value of $f(z^*) = 0.012656987$. The results reveal that EPO again performs better than the other optimization algorithms. The statistical results of tension/compression spring design are compared and tabulated in Table 20. From Table 20, it can be seen that provides better statistical results than the other algorithms in terms of best, mean, and median.

The convergence analysis of best optimal solution obtained from EPO is shown in Fig. 21.

5.1.5. 25-bar truss design

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

- 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 21 shows the member stress limitations for this truss problem. The loading conditions for 25-bar truss is presented in Table 22. The comparison of best solutions among several algorithms is provided in Table 23. The proposed EPO is better than other competitor algorithms in terms of best, average, and standard deviation. EPO converges very efficiently to optimize this problem as shown in Fig. 22.

5.1.6. 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}, ε, e, and ζ (see Fig. 23). The mathematical representation of this problem is given below:

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

Table 20
Statistical results obtained from different algorithms for tension/compression spring design problem.

Algorithms	Best	Mean	Worst	Std. Dev.	Median
EPO	**0.012656987**	**0.012678903**	0.012667902	0.001021	**0.012676002**
SHO	0.012674000	0.012684106	0.012715185	**0.000027**	0.012687293
GWO	0.012678321	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
HS	0.012776352	0.013069872	0.015214230	0.000375	0.012952142

Table 21
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)

Table 22
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	10.0 (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 23
Statistical results obtained from different algorithms for 25-bar truss design problem.

Groups	EPO	ACO[66]	PSO[67]	CSS[68]	BB-BC[69]
$A1$	**0.01**	0.01	0.01	0.01	0.01
$A2-A5$	**1.850**	2.042	2.052	2.003	1.993
$A6-A9$	**3.001**	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
$A14-A17$	**0.660**	0.684	0.684	0.687	0.665
$A18-A21$	1.621	1.625	**1.616**	1.655	1.642
$A22-A25$	2.671	2.672	2.673	**2.66**	2.679
Best weight	**544.92**	545.03	545.21	545.10	545.16
Average weight	**545.13**	545.74	546.84	545.58	545.66
Std. dev.	**0.397**	0.94	1.478	0.412	0.491

Subject to:

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

where,

$$f_c = 37.91\left[1 + \left\{1.04\left(\frac{1-\gamma}{1+\gamma}\right)^{1.72}\left(\frac{f_i(2f_o-1)}{f_o(2f_i-1)}\right)^{0.41}\right\}^{10/3}\right]^{-0.3} \times \left[\frac{\gamma^{0.3}(1-\gamma)^{1.39}}{(1+\gamma)^{1/3}}\right]\left[\frac{2f_i}{2f_i-1}\right]^{0.41} \quad (18)$$

$$x = [\{(D-d)/2 - 3(T/4)\}^2 + \{D/2 - T/4 - D_b\}^2 - \{d/2 + T/4\}^2]$$
$$y = 2\{(D-d)/2 - 3(T/4)\}\{D/2 - T/4 - D_b\}$$
$$\phi_o = 2\pi - 2cos^{-1}\left(\frac{x}{y}\right)$$
$$\gamma = \frac{D_b}{D_m}, \quad f_i = \frac{r_i}{D_b}, \quad f_o = \frac{r_o}{D_b}, \quad T = D - d - 2D_b$$
$$D = 160, \quad d = 90, \quad B_w = 30, \quad r_i = r_o = 11.033$$
$$0.5(D+d) \le D_m \le 0.6(D+d), \quad 0.15(D-d) \le D_b \le 0.45(D-d), \quad 4 \le Z \le 50, \quad 0.515 \le f_i and f_o \le 0.6,$$
$$0.4 \le K_{Dmin} \le 0.5, \quad 0.6 \le K_{Dmax} \le 0.7, \quad 0.3 \le e \le 0.4,$$
$$0.02 \le e \le 0.1, \quad 0.6 \le \zeta \le 0.85.$$

Table 24 shows the performance comparison of best optimal solution obtained from several algorithms. The proposed EPO provides optimal solution at z^*=(125, 21.41890, 10.94113, 0.515, 0.515, 0.4, 0.7, 0.3, 0.02, 0.6) with corresponding fitness value equal to $f(z^*) = 85067.983$. The statistical results obtained for rolling element bearing design problem are compared and shown in Table 25.

Fig. 24 shows the convergence analysis of EPO algorithm. It reveals that EPO is capable to achieve a best optimal solution.

Table 24
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}	ε	e	ζ	
EPO	125	21.41890	10.94113	0.515	0.515	0.4	0.7	0.3	0.02	0.6	**85067.983**
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	11.17342	0.515	0.515000	0.5	0.61503	0.300000	0.05161	0.60000	81691.202
MVO	125.6002	21.32250	10.97338	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	11.14989	0.515	0.517746	0.5	0.61827	0.304068	0.02000	0.624638	82276.941
GA	125	20.77562	11.01247	0.515	0.515000	0.5	0.61397	0.300000	0.05004	0.610001	82773.982
HS	125	20.87123	11.16697	0.515	0.516000	0.5	0.61951	0.301128	0.05024	0.614531	81569.527

Table 25
Statistical results obtained from different algorithms for Rolling element bearing design problem.

Algorithms	Best	Mean	Worst	Std. Dev.	Median
EPO	**85067.983**	**85042.352**	86551.599	1877.09	**85056.095**
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**	**13962.150**	42287.581
MVO	84491.266	84353.685	84100.834	0392.431	84398.601
SCA	83431.117	81005.232	77992.482	1710.777	81035.109
GSA	82276.941	78002.107	71043.110	3119.904	78398.853
GA	82773.982	81198.753	80687.239	1679.367	8439.728
HS	81569.527	80397.998	79412.779	1756.902	8347.009

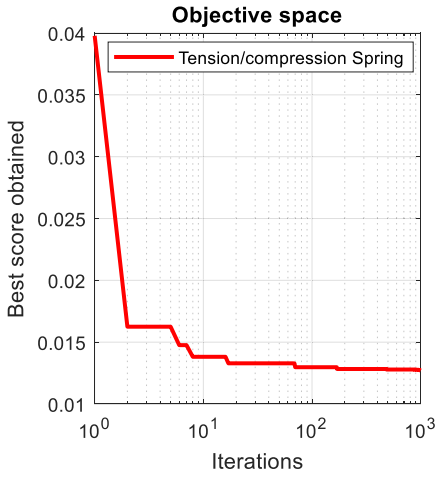

Fig. 21. Convergence analysis of EPO for tension/compression spring design problem.

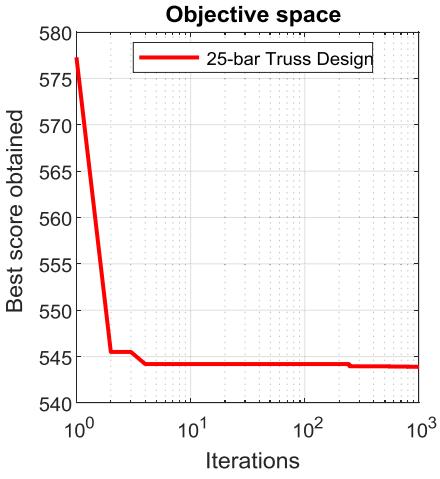

Fig. 22. Convergence analysis of EPO for 25-bar truss design problem.

5.2. Unconstrained design problem

Unconstrained optimization problems are such problems that depends on the real variable without any restrictions on their values. This section describes the unconstrained displacement of loaded structure design problem to minimize the potential energy.

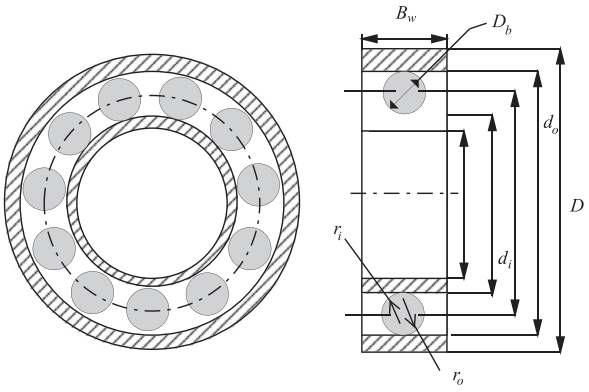

Fig. 23. Schematic view of rolling element bearing problem.

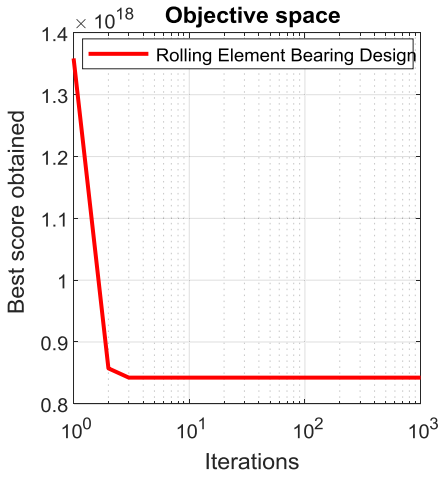

Fig. 24. Convergence analysis of EPO for rolling element bearing design problem.

5.2.1. Displacement of loaded structure

A displacement is a vector which defines the shortest distance between the initial and the final position of a given point.

The main objective of this problem is to minimize the potential energy for reducing the excess load of structure. Fig. 14 shows the loaded structure that should have minimum potential energy ($f(\vec{z})$). The problem can be stated as follows:

$$
\begin{aligned}
&f(\vec{z}) = \text{Minimize}_{z_1,z_2}\pi \\
&\text{where,} \\
&\pi = \frac{1}{2}K_1u_1^2 + \frac{1}{2}K_2u_2^2 - F_zz_1 - F_yz_2 \\
&K_1 = 8N/cm,\ K_2 = 1N/cm,\ F_y = 5N,\ F_z = 5N \\
&u_1 = \sqrt{z_1^2 + (10 - z_2^2)} - 10, \quad u_2 = \sqrt{z_1^2 + (10 + z_2^2)} - 10.
\end{aligned} \tag{19}
$$

Table 26 shows the comparison of best optimal solution obtained from EPO and other competitor approaches such as SHO, GWO, PSO, MVO, SCA, GSA, GA, and HS. The proposed EPO provides best optimum cost at $\pi = 168.8231$. The results reveal that EPO is able to minimize the potential energy for loaded structure problem.

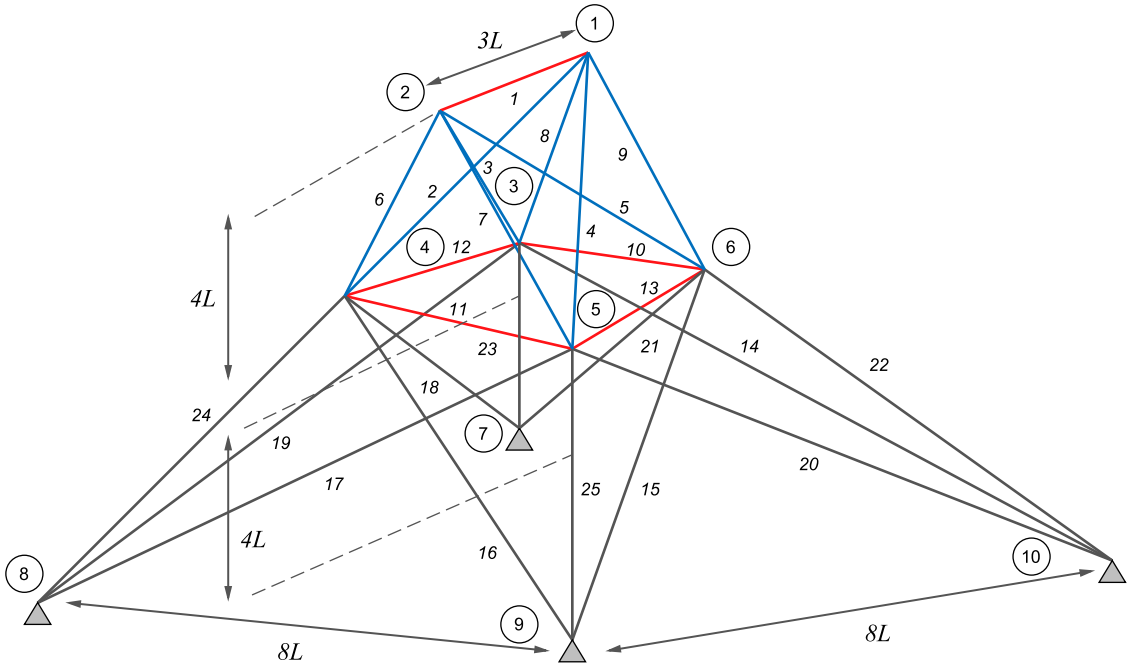

Fig. 25. Schematic view of 25-bar truss problem.

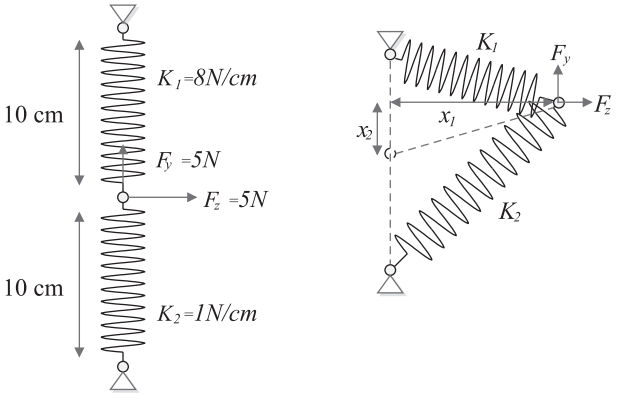

Fig. 26. Schematic view of displacement of loaded structure.

Table 26
Comparison results for displacement of loaded structure problem.

Algorithms	Optimum cost (π)
EPO	**168.8231**
SHO	168.8889
GWO	170.3645
PSO	170.5960
MVO	169.3023
SCA	169.0032
GSA	176.3697
GA	171.3674
HS	172.0324

Table 27
Statistical results obtained from different algorithms for displacement of loaded structure problem.

Algorithms	Best	Mean	Worst	Std. Dev.	Median
EPO	**168.8231**	**170.1309**	230.9721	211.861	**169.4214**
SHO	168.8889	170.3659	173.6357	**023.697**	169.6710
GWO	170.3645	171.3694	174.3970	196.037	173.3694
PSO	170.5960	174.6354	175.3602	236.036	173.9634
MVO	169.3023	171.0034	174.3047	202.753	170.0032
SCA	169.0032	171.7530	174.4527	129.047	170.3647
GSA	176.3697	178.7521	179.5637	113.037	174.367
GA	171.3674	172.0374	174.0098	212.703	172.0097
HS	172.0324	173.4327	174.0399	080.111	171.3697

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

(a_{ij}, i = 1, 2 and j = 1, 2, ...,25)								
i\j	1	2	3	4	5	6	...	25
1	−32	−16	0	16	32	−32	...	32
2	−32	−32	−32	−32	−32	−16	...	32

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

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

The statistical results for the reported algorithms are depicted in Table 27. From Table 27, it is noticed that the results obtained from EPO are far better than the other algorithms in terms of best, mean, and median.

Fig. 27 shows the convergence analysis of best solution obtained from EPO.

In summary, these results reveal that EPO is an effective optimizer for solving both constrained and unconstrained engineering design problems with low computational cost and fast convergence speed.

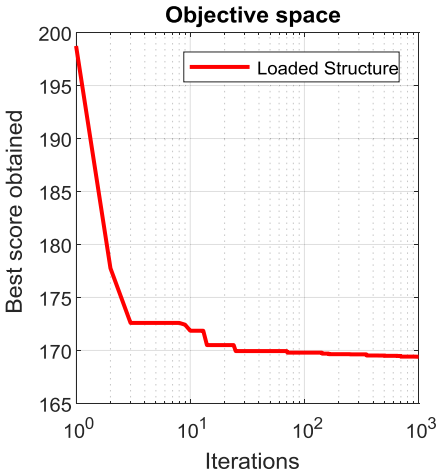

Fig. 27. Convergence analysis of EPO for displacement of loaded structure problem.

6. Conclusion

This paper presents a novel swarm-based metaheuristic algorithm called Emperor Penguin Optimizer (EPO). The fundamental concept behind this algorithm is the huddling behavior of emperor penguins. The proposed EPO algorithm has been tested on forty-four benchmark test functions. Moreover, seven real-life engineering design problems are employed to further determine the efficiency of proposed algorithm. The results reveal that EPO provides very competitive results as compared with other well-known metaheuristics such as SHO, GWO, PSO, MVO, SCA, GSA, GA, and HS. Despite, the computational complexity has also been analyzed in terms of time and space complexity. The statistical results, based on the comparison of proposed algorithm against other optimization approaches, show that EPO handles various types of constraints very efficiently and provides better solutions.

The results on the unimodal and multimodal test functions shows the superior exploitation and exploration capability of EPO algorithm, respectively. The results of the composite benchmark functions shows the high local optima avoidance of proposed algorithm. Finally, the algorithm is tested on very challenging special session with bound constraints CEC 2015 benchmark test functions.

Moreover, the proposed algorithm is applicable on seven real-life optimization problems to show its efficiency in a given search space.

There are several research directions which can be recommended for future works. The binary version of EPO algorithm can be seen as a future contribution. Also, to extend this algorithm for solving multi-objective as well as many-objective real-life optimization problems.

Table 31
Hartman Function F_{20}.

i	$(a_{ij}, j = 1, 2, ..., 6)$						c_i	$(p_{ij}, j = 1, 2, ..., 6)$					
1	10	3	17	3.5	1.7	8	1	0.1312	0.1696	0.5569	0.0124	0.8283	0.5886
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	0.3047	0.6650
4	17	8	0.05	10	0.1	14	3.2	0.4047	0.8828	0.8732	0.5743	0.1091	0.0381

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

A1. Unimodal benchmark test functions

A1.1. Sphere model

$F_1(z) = \sum_{i=1}^{30} z_i^2$

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

A1.2. Schwefel'S problem 2.22

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

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

A1.3. Schwefel'S problem 1.2

$F_3(z) = \sum_{i=1}^{30} \left(\sum_{j=1}^{i} z_j \right)^2 - 100 \le z_i \le 100, \quad f_{\min} = 0, \quad Dim = 30$

A1.4. Schwefel'S problem 2.21

$F_4(z) = max_i\{|z_i|, 1 \le i \le 30\} - 100 \le z_i \le 100, \quad f_{\min} = 0, \quad Dim = 30$

A1.5. Generalized Rosenbrock's function

$F_5(z) = \sum_{i=1}^{29} [100(z_{i+1} - z_i^2)^2 + (z_i - 1)^2] - 30 \le z_i \le 30, \quad f_{\min} = 0, \quad Dim = 30$

A1.6. Step function

$F_6(z) = \sum_{i=1}^{30} (|z_i + 0.5|)^2 - 100 \le z_i \le 100, \quad f_{\min} = 0, \quad Dim = 30$

A1.7. Quartic function

$F_7(z) = \sum_{i=1}^{30} iz_i^4 + random[0, 1] - 1.28 \le z_i \le 1.28, \quad f_{\min} = 0, \quad Dim = 30$

A2. Multimodal benchmark test functions

A2.1. Generalized Schwefel's problem 2.26

$F_8(z) = \sum_{i=1}^{30} -z_i \sin(\sqrt{|z_i|}) - 500 \le z_i \le 500, \quad f_{\min} = -12569.5, \quad Dim = 30$

A2.2. Generalized Rastrigin's function

$F_9(z) = \sum_{i=1}^{30} [z_i^2 - 10cos(2\pi z_i) + 10] - 5.12 \le z_i \le 5.12, \quad f_{\min} = 0, \quad Dim = 30$

A2.3. Ackley's function

$F_{10}(z) = -20exp\left(-0.2\sqrt{\frac{1}{30}\sum_{i=1}^{30} z_i^2}\right) - \exp\left(\frac{1}{30}\sum_{i=1}^{30} \cos(2\pi z_i)\right) + 20 + e - 32 \le z_i \le 32, \quad f_{\min} = 0, \quad Dim = 30$

A2.4. Generalized griewank function

$F_{11}(z) = \frac{1}{4000}\sum_{i=1}^{30} z_i^2 - \prod_{i=1}^{30} \cos\left(\frac{z_i}{\sqrt{i}}\right) + 1 - 600 \le z_i \le 600, \quad f_{\min} = 0, \quad Dim = 30$

A2.5. Generalized penalized functions

- $F_{12}(z) = \frac{\pi}{30}\{10sin(\pi x_1) + \sum_{i=1}^{29}(x_i - 1)^2[1 + 10sin^2(\pi x_{i+1})] + (x_n - 1)^2\} - 50 \le z_i \le 50, \quad f_{\min} = 0, \quad Dim = 30$
  $+ \sum_{i=1}^{30} u(z_i, 10, 100, 4)$
- $F_{13}(z) = 0.1\{sin^2(3\pi z_1) + \sum_{i=1}^{29}(z_i - 1)^2[1 + \sin^2(3\pi z_i + 1)]$
  $+ (z_n - 1)^2[1 + \sin^2(2\pi z_{30})]\} + \sum_{i=1}^{N} u(z_i, 5, 100, 4)$

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

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

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

A3. Fixed-dimension multimodal benchmark test functions

A3.1. Shekel's foxholes function

$F_{14}(z) = \left(\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(z_i - a_{ij})^6}\right)^{-1} - 65.536 \le z_i \le 65.536, \quad f_{min} \approx 1, \quad Dim = 2$

A3.2. Kowalik's function

$F_{15}(z) = \sum_{i=1}^{11} \left[a_i - \frac{z_1(b_i^2 + b_i z_2)}{b_i^2 + b_i z_3 + z_4}\right]^2 - 5 \le z_i \le 5, \quad f_{min} \approx 0.0003075, \quad Dim = 4$

A3.3. Six-Hump camel-Back function

$$F_{16}(z) = 4z_1^2 - 2.1z_1^4 + \frac{1}{3}z_1^6 + z_1z_2 - 4z_2^2 + 4z_2^4 \quad -5 \le z_i \le 5, \quad f_{min} = -1.0316285, \quad Dim = 2$$

A3.4. Branin function

$$F_{17}(z) = \left(z_2 - \frac{5.1}{4\pi^2}z_1^2 + \frac{5}{\pi}z_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)cosz_1 + 10$$
$$-5 \le z_1 \le 10, \quad 0 \le z_2 \le 15, \quad f_{min} = 0.398, \quad Dim = 2$$

A3.5. Goldstein-price function

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

A3.6. Hartman's family

- $F_{19}(z) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}(z_j - p_{ij})^2\right)$ $0 \le z_j \le 1, \quad f_{min} = -3.86, \quad Dim = 3$
- $F_{20}(z) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}(z_j - p_{ij})^2\right)$ $0 \le z_j \le 1, \quad f_{min} = -3.32, \quad Dim = 6$

A3.7. Shekel's foxholes function

- $F_{21}(z) = -\sum_{i=1}^{5} [(X - a_i)(X - a_i)^T + c_i]^{-1}$
  $0 \le z_i \le 10, \quad f_{min} = -10.1532, \quad Dim = 4$
- $F_{22}(z) = -\sum_{i=1}^{7} [(X - a_i)(X - a_i)^T + c_i]^{-1}$
  $0 \le z_i \le 10, \quad f_{min} = -10.4028, \quad Dim = 4$
- $F_{23}(z) = -\sum_{i=1}^{10} [(X - a_i)(X - a_i)^T + c_i]^{-1}$
  $0 \le z_i \le 10, \quad f_{min} = -10.536, \quad Dim = 4$

A4. Basic composite benchmark test functions

A4.1. Weierstrass function

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

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.

A5. Basic CEC 2015 benchmark test functions

A5.1. Bent cigar function

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

A5.2. Discus function

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

A5.3. Modified Schwefel's function

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

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

A5.4. Katsuura function

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

A5.5. Happycat function

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

A5.6. HGBat Function

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

A5.7. Expanded Griewank's plus Rosenbrock's function

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

A5.8. Expanded Scaffer's F6 function

Scaffer's *F*6 Function:

$g(z, x) = 0.5 + \frac{(sin^2(\sqrt{z^2 + x^2}) - 0.5)}{(1 + 0.001(z^2 + x^2))^2} F(z) = g(z_1, z_2) + g(z_2, z_3) + ...+g(z_{30}, z_1)$

A5.9. 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 10.

Table A.4
Composite benchmark test functions.

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

Table A.5
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 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 Rastrigin's Function High Conditioned Elliptic Function	30	1000
$CEC - 11$	Hybrid Function 2 ($N = 4$)	Griewank's Function Weierstrass Function Rosenbrock's Function Scaffer's F6 Function	30	1100
$CEC - 12$	Hybrid Function 3 ($N = 5$)	Katsuura Function HappyCat Function Expanded Griewank's plus Rosenbrock's Function Schwefel's Function Ackley's Function	30	1200
$CEC - 13$	Composition Function 1 ($N = 5$)	Rosenbrock's Function High Conditioned Elliptic Function Bent Cigar Function Discus Function High Conditioned Elliptic Function	30	1300
$CEC - 14$	Composition Function 2 ($N = 3$)	Schwefel's Function Rastrigin's Function High Conditioned Elliptic Function	30	1400
$CEC - 15$	Composition Function 3 ($N = 5$)	HGBat Function Rastrigin's Function Schwefel's Function Weierstrass Function High Conditioned Elliptic Function	30	1500

References

[1] R. K. Chandrawat, R. Kumar, B. P. Garg, G. Dhiman, S. Kumar, An Analysis of Modeling and Optimization Production Cost Through Fuzzy Linear Programming Problem with Symmetric and Right Angle Triangular Fuzzy Number, Springer Singapore, Singapore, pp. 197–211.

[2] P. Singh, G. Dhiman, A fuzzy-lp approach in time series forecasting, in: B.U. Shankar, K. Ghosh, D.P. Mandal, S.S. Ray, D. Zhang, S.K. Pal (Eds.), Pattern Recognition and Machine Intelligence, Springer International Publishing, Cham, 2017, pp. 243–253.

[3] P. Singh, G. Dhiman, Uncertainty representation using fuzzy-entropy approach: special application in remotely sensed high resolution satellite images (RSHRSIs), Appl. Soft Comput. (2018) in press.

[4] E. Alba, B. Dorronsoro, The exploration/exploitation tradeoff in dynamic cellular genetic algorithms, IEEE Trans. Evolut. Comput. 9 (2) (2005) 126–142, http://dx.doi.org/10.1109/TEVC.2005.843751.

[5] O. Olorunda, A.P. Engelbrecht, Measuring exploration/exploitation in particle swarms using swarm diversity, IEEE Congr. Evolut. Comput. (2008) 1128–1134, http://dx.doi.org/10.1109/CEC.2008.4630938.

[6] M. Lozano, C. Garcia-Martinez, Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: overview and progress report, Comput. Oper. Res. 37(3) (2010) 481–497, http://dx.doi.org/10.1016/j.cor.2009.02.010.

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

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

[9] E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, GSA: a gravitational search algorithm, Inf. Sci. (NY) 179 (13) (2009) 2232–2248, http://dx.doi.org/10.1016/j.ins.2009.03.004.

[10] O.K. Erol, I. Eksin, A new optimization method: big bang-big crunch, Adv. Eng. Softw. 37 (2) (2006) 106–111, http://dx.doi.org/10.1016/j.advengsoft.2005.04.005.

[11] A. Kaveh, S. Talatahari, A novel heuristic optimization method: charged system search, Acta Mech. 213 (3) (2010) 267–289, http://dx.doi.org/10.1007/s00707-009-0270-4.

[12] A. Hatamlou, Black hole: a new heuristic optimization approach for data clustering, Inf. Sci. (NY) 222 (2013) 175–184, http://dx.doi.org/10.1016/j.ins.2012.08.023.

[13] R.A. Formato, Central force optimization: a new deterministic gradient-like optimization metaheuristic, Opsearch 46 (1) (2009) 25–51, http://dx.doi.org/10.1007/s12597-009-0003-4.

[14] H. Du, X. Wu, J. Zhuang, Small-World Optimization Algorithm for Function Optimization, Springer, Berlin Heidelberg, 2006, pp. 264–273.

[15] B. Alatas, ACROA: artificial chemical reaction optimization algorithm for global optimization, Expert Syst. Appl. 38 (10) (2011) 13170–13180, http://dx.doi.org/10.1016/j.eswa.2011.04.126.

[16] A. Kaveh, M. Khayatazad, A new meta-heuristic method: ray optimization, Comput. Struct. 112–113 (2012) 283–294, http://dx.doi.org/10.1016/j.compstruc.2012.09.003.

[17] H. Shah Hosseini, Principal components analysis by the galaxy-based search algorithm: a novel metaheuristic for continuous optimisation, Int. J. Comput. Sci. Eng. 6 (2011) 132–140, http://dx.doi.org/10.1504/IJCSE.2011.041221.

[18] Moghaddam FF, Moghaddam RF, Cheriet M. Curved space optimization: A random search based on general relativity theory, 2012. arXiv preprint arXiv:1208.2214.

[19] J.H. Holland, Genetic algorithms, Sci. Am. 267 (1) (1992) 66–72.

[20] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (4) (1997) 341–359, http://dx.doi.org/10.1023/A:1008202821328.

[21] J.R. Koza, Genetic Programming: on the Programming of Computers by Means of Natural Selection, MIT Press, 1992.

[22] H.-G. Beyer, H.-P. Schwefel, Evolution strategies–a comprehensive introduction, Nat. Comput.s 1 (1) (2002) 3–52, http://dx.doi.org/10.1023/A:1015059928466.

[23] D. Simon, Biogeography-based optimization, IEEE Trans. Evolut. Comput. 12 (6)

(2008) 702–713, http://dx.doi.org/10.1109/TEVC.2008.919004.
[24] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceedings of the IEEE International Conference on Neural Networks, 1995, pp. 1942–1948.
[25] A. Slowik, H. Kwasnicka, Nature inspired methods and their industry applications - swarm intelligence algorithms, IEEE Trans. Ind. Inf. PP (99) (2017). 1–1. doi:10.1109/TII.2017.2786782
[26] M. Dorigo, M. Birattari, T. Stutzle, Ant colony optimization - artificial ants as a computational intelligence technique, IEEE Comput. Intell. Mag. 1 (2006) 28–39.
[27] X.-S. Yang, A New Metaheuristic Bat-Inspired Algorithm, Springer, Berlin Heidelberg, 2010, pp. 65–74.
[28] D. Karaboga, B. Basturk, Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems, Springer, Berlin Heidelberg, Berlin, Heidelberg, pp. 789–798.
[29] G. Dhiman, V. Kumar, Spotted hyena optimizer: a novel bio-inspired based meta-heuristic technique for engineering applications, Adv. Eng. Softw. 114 (2017) 48–70, http://dx.doi.org/10.1016/j.advengsoft.2017.05.014.
[30] X.S. Yang, S. Deb, Cuckoo search via levy flights, in: Proceedings of the World Congress on Nature Biologically Inspired Computing, 2009, pp. 210–214. 10.1109/NABIC.2009.5393690.
[31] A. Mucherino, O. Seref, Monkey search: a novel metaheuristic search for global optimization, AIP Conf. Proc. 953 (1) (2007) 162–173.
[32] S. Das, A. Biswas, S. Dasgupta, A. Abraham, Bacterial Foraging Optimization Algorithm: Theoretical Foundations, Analysis, and Applications, Springer, Berlin, Heidelberg, pp. 23–55.
[33] X.-S. Yang, Firefly algorithm, stochastic test functions and design optimisation, Int. J. Bio Inspired Comput. 2 (2) (2010) 78–84, http://dx.doi.org/10.1504/IJBIC.2010.032124.
[34] W.-T. Pan, A new fruit fly optimization algorithm: taking the financial distress model as an example, Knowl. Based Syst. 26 (2012) 69–74.
[35] L. Wu, Q. Liu, X. Tian, J. Zhang, W. Xiao, A new improved fruit fly optimization algorithm IAFOA and its application to solve engineering optimization problems, Knowl. Based Syst. 144 (2018) 153–173, http://dx.doi.org/10.1016/j.knosys.2017.12.031.
[36] X. Han, Q. Liu, H. Wang, L. Wang, Novel fruit fly optimization algorithm with trend search and co-evolution, Knowl. Based Syst. 141 (2018) 1–17, http://dx.doi.org/10.1016/j.knosys.2017.11.001.
[37] M. Mitic, N. Vukovic, M. Petrovic, Z. Miljkovic, Chaotic fruit fly optimization algorithm, Knowl. Based Syst. 89 (2015) 446–458, http://dx.doi.org/10.1016/j.knosys.2015.08.010.
[38] Y. Wang, S. Wu, D. Li, S. Mehrabi, H. Liu, A part-of-speech term weighting scheme for biomedical information retrieval, J. Biomed. Inform. 63 (2016) 379–389, http://dx.doi.org/10.1016/j.jbi.2016.08.026.
[39] C. Orozco-Henao, A. Bretas, R. Chouhy-Leborgne, A. Herrera-Orozco, J. Marin-Quintero, Active distribution network fault location methodology: a minimum fault reactance and fibonacci search approach, Int. J. Elect. Power Energy Syst. 84 (2017) 232–241, http://dx.doi.org/10.1016/j.ijepes.2016.06.002.
[40] A. Askarzadeh, Bird mating optimizer: an optimization algorithm inspired by bird mating strategies, Commun. Nonlinear Sci. Numer. Simul. 19 (4) (2014) 1213–1228.
[41] A.H. Gandomi, A.H. Alavi, Krill herd: a new bio-inspired optimization algorithm, Commun. Nonlinear Sci. Numer. Simul. 17 (12) (2012) 4831–4845.
[42] M. Neshat, G. Sepidnam, M. Sargolzaei, A.N. Toosi, Artificial fish swarm algorithm: a survey of the state-of-the-art, hybridization, combinatorial and indicative applications, Artif. Intell. Rev. 42 (4) (2014) 965–997.
[43] Y. Shiqin, J. Jianjun, Y. Guangxing, A dolphin partner optimization, in: Proceedings of the WRI Global Congress on Intelligent Systems (2009) 124–128. 10.1109/GCIS.2009.464.
[44] R. Oftadeh, M. Mahjoob, M. Shariatpanahi, A novel meta-heuristic optimization algorithm inspired by group hunting of animals: hunting search, Comput. Math. Appl. 60 (7) (2010) 2087–2098 https://doi.org/10.1016/j.camwa.2010.07.049.
[45] S. Mirjalili, Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm, Knowl. Based Syst. 89 (2015) 228–249.
[46] G. Dhiman, A. Kaur, A hybrid algorithm based on particle swarm and spotted hyena optimizer for global optimization, Advances in Intelligent Systems and Computing, Springer, 2018. in press
[47] G. Dhiman, V. Kumar, Spotted hyena optimizer for solving complex and non-linear constrained engineering problems, Advances in Intelligent Systems and Computing, Springer, 2018. in press
[48] A. Kaur, G. Dhiman, A review on search based tools and techniques to identify bad code smells in object oriented systems, Advances in Intelligent Systems and Computing, Springer, 2018. in press
[49] P. Singh, G. Dhiman, A hybrid fuzzy time series forecasting model based on granular computing and bio-inspired optimization approaches, J. Comput. Sci. (2018), http://dx.doi.org/10.1016/j.jocs.2018.05.008.
[50] A. Waters, F. Blanchette, A.D. Kim, Modeling huddling penguins, PLoS ONE 7 (11) (2012) e50277.
[51] J. Digalakis, K. Margaritis, On benchmarking functions for genetic algorithms, Int. J. Comput. Math. 77 (4) (2001) 481–506, http://dx.doi.org/10.1080/00207160108805080.
[52] J.J. Liang, P.N. Suganthan, K. Deb, Novel composition test functions for numerical global optimization, in: Proceedings of the IEEE Swarm Intelligence Symposium, 2005, pp. 68–75. 10.1109/SIS.2005.1501604.
[53] Q. Chen, B. Liu, Q. Zhang, J. Liang, P. Suganthan, B. Qu, Problem definitions and evaluation criteria for CEC 2015 special session on bound constrained single-objective computationally expensive numerical optimization, Technical Report, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University (2014).
[54] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y.-P. Chen, A. Auger, S. Tiwari, Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization, Technical Report, Nanyang Technological University, Singapore (2005).
[55] G. Dhiman, A. Kaur, Spotted hyena optimizer for solving engineering design problems, in: Proceedings of the International Conference on Machine Learning and Data Science (MLDS), 2017, pp. 114–119. 10.1109/MLDS.2017.5.
[56] G. Dhiman, V. Kumar, Multi-objective spotted hyena optimizer: a multi-objective optimization algorithm for engineering problems, Knowl. Based Syst. 150 (2018) 175–197, http://dx.doi.org/10.1016/j.knosys.2018.03.011.
[57] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer, Adv. Eng. Softw. 69 (2014) 46–61. http://doi.org/10.1016/j.advengsoft.2013.12.007 .
[58] S. Mirjalili, S.M. Mirjalili, A. Hatamlou, Multi-verse optimizer: a nature-inspired algorithm for global optimization, Neural Comput. Appl. 27 (2) (2016) 495–513, http://dx.doi.org/10.1007/s00521-015-1870-7.
[59] S. Mirjalili, SCA: a sine cosine algorithm for solving optimization problems, Knowl. Based Syst. 96 (2016) 120–133 https://doi.org/10.1016/j.knosys.2015.12.022.
[60] E. Bonabeau, M. Dorigo, G. Theraulaz, Swarm Intelligence: From Natural to Artificial systems, Oxford University Press, Inc., 1999.
[61] Z.W. Geem, J.H. Kim, G.V. Loganathan, A new heuristic optimization algorithm: harmony search, Simulation 76 (2) (2001) 60–68, http://dx.doi.org/10.1177/003754970107600201.
[62] C.A.C. Coello, Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art, Comput. Methods Appl. Mech. Eng. 191 (11–12) (2002) 1245–1287 https://doi.org/10.1016/S0045-7825(01)00323-1.
[63] B. Kannan, S.N. Kramer, An augmented lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design, J. Mech. Des. 116 (2) (1994) 405–411.
[64] A.H. Gandomi, X.-S. Yang, Benchmark Problems in Structural Optimization, Springer, Berlin Heidelberg, 2011, pp. 259–281.
[65] E. Mezura-Montes, C.A.C. Coello, Useful Infeasible Solutions in Engineering Optimization with Evolutionary Algorithms, Springer, Berlin Heidelberg, 2005, pp. 652–662.
[66] C.V.C.B.J. Bichon, Design of space trusses using ant colony optimization, J. Struct. Eng. 130 (5) (2004) 741–751.
[67] J. Schutte, A. Groenwold, Sizing design of truss structures using particle swarms, Struct. Multidiscip. Optim. 25 (4) (2003) 261–269, http://dx.doi.org/10.1007/s00158-003-0316-5.
[68] A. Kaveh, S. Talatahari, Optimal design of skeletal structures via the charged system search algorithm, Struct. Multidiscip. Optim. 41 (6) (2010) 893–911, http://dx.doi.org/10.1007/s00158-009-0462-5.
[69] A. Kaveh, S. Talatahari, Size optimization of space trusses using big bang-big crunch algorithm, Comput. Struct. 87 (17–18) (2009) 1129–1140, http://dx.doi.org/10.1016/j.compstruc.2009.04.011.