Applied Soft Computing Journal 73 (2018) 697–726

Contents lists available at ScienceDirect

Applied Soft Computing Journal

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

PSOSCALF: A new hybrid PSO based on Sine Cosine Algorithm and Levy flight for solving optimization problems

Saeed Nezamivand Chegini *, Ahmad Bagheri, Farid Najafi

Department of Mechanical Engineering, Faculty of Engineering, University of Guilan, Rasht, Iran

HIGHLIGHTS

- This paper proposes a new hybrid optimization algorithm called PSOSCALF.
- Performance is tested using benchmark functions and constrained engineering problems.
- Proposed algorithm is compared with well-known optimization algorithm.
- The results show that the PSOSCALF is more effective than other methods.
- Superiority of PSOSCALF is also proved by applying on engineering design problems.

ARTICLE INFO

Article history:
Received 27 October 2017
Received in revised form 13 August 2018
Accepted 18 September 2018
Available online 27 September 2018

Keywords:
Particle swarm optimization (PSO)
Levy flight distribution
Sine Cosine algorithm (SCA)
Exploration
Exploitation
Function optimization
Constrained optimization

ABSTRACT

The development of the meta-heuristic algorithms for solving the optimization problems and constrained engineering problems is one of the topics of interest to researchers in recent years. Particle swarm optimization algorithm (PSO) is one of the social search-based and swarm intelligence algorithms that is distinguished by its high speed, low number of parameters and easy implementation. However, the PSO algorithm has disadvantages such as finding the local minimum instead of the global minimum and debility in global search capability. In this article, in order to solve these deficiencies, the PSO algorithm is combined with position updating equations in Sine Cosine Algorithm (SCA) and the Levy flight approach. Therefore, a new hybrid method called PSOSCALF is introduced in this paper. In the SCA algorithm, the mathematical formulation for the solution updating is based on the behavior of sine and cosine functions. These functions guarantee the exploitation and exploration capabilities. Levy flight is a random walk that produces search steps using Levy distribution and then, with large jumps, more effective searches are occurred in the search space. Thus, using combination of the SCA and Levy flight in the PSOSCALF algorithm, the exploration capability of the original PSO algorithm is enhanced and also, being trapped in the local minimum is prevented. The performance and accuracy of the PSOSCALF method have been examined by 23 benchmark functions of the unimodal and multimodal type and 8 constrained real problems in engineering. The optimization results of the test functions show that the PSOSCALF method is more successful than the PSO family and other algorithms in determining global minimum of these functions. Also, the proposed PSOSCALF algorithm is successfully applied to the real constrained engineering problems and provides better solutions than other methods.

1. Introduction

In recent years, different optimization algorithm are used for solving the real engineering problems. These methods have been inspired by the natural facts. Unlike classical algorithms, these algorithms are useful in solving optimization problems that are non-differentiable and non-continues. The principles applied in most of the meta-heuristic algorithms have been obtained by studying the behavior of the creatures in nature [1–3]. For example: artificial bee colony optimization (ABC) algorithm is based on colony behavior of the bees [4], the ant colony optimization (ACO) algorithm that simulates the actual behavior of ants [5], the whale optimization algorithm (WOA) emulates the behavior of humpback whales in hunting [6], the cuckoo search optimization algorithm was inspired by the parasitic life of some cuckoo bird species in nesting and laying to solve optimization problems [7] and the particle swarm optimization algorithm (PSO) is one of the swarm intelligence methods that its principles follow social behavior of animals such as fish and bird [8].

Having two features of exploration and exploitation and the balance between these two features in a meta-heuristic algorithm will empower the algorithm to find the global optimum. In the

* Corresponding author.
E-mail address: saeed.nezamivand@gmail.com (S.N. Chegini).

https://doi.org/10.1016/j.asoc.2018.09.019
1568-4946/

exploration stage, the algorithm attempts to search the space for finding the new response. In fact, in this phase, the chance of the determining the global optimum is increased. But, in the exploitation phase, the algorithm searches neighborhood of the highest quality solution that obtained so far and chooses the best solution. The optimization algorithm with the best efficiency has two features as follows: in the inception of the process of finding the optimal solution, has the exploration ability and in the final stages of the optimization process, has the exploitation ability.

The principles of the PSO optimization algorithm are based on population, motion and intelligence of individuals. In fact, this algorithm creates new solutions for optimization problems using the concept of the swarm intelligence. The number of adjustable parameters in this algorithm is low and as a result, implementation of this algorithm is very simple. These features have attracted researchers in recent years in various scientific fields and engineering issues, such as function optimization [9], aircraft wing design [10], rotating machinery fault detection [11], signal de-noising [12], controller design [13] and nonlinear system identification [14], etc.

However, various studies show that in addition to the advantages of the PSO algorithm, there are two main difficulties in this algorithm: (1) trapping in the local minimum instead of finding newer solutions and improving them and (2) early convergence. To overcome these problems, various types of particle swarm optimization algorithm have been developed. Shi and Eberhart [15] found that by changing the value of the inertia weight in the particle velocity equation, the global search ability for the PSO algorithm is increased and high quality solutions are generated. In the proposed method in this paper, the value of the inertia weight linearly decreases from 0.9 to 0.4 during the optimization process. Ratnaweera et al. [16] introduced a new method based on the concept "self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients (HPSO-TVAC)". In the HPSO-TAVC method, the new velocity of each particle is calculated without considering the previous velocity vector term. Then, if the new calculated velocity is equal to zero, in this case, the velocity of the particle is generated with a time-varying reinitialization velocity strategy. In other words, in their proposed method, the velocity of each particle is estimated based on the "social" part and the "cognitive" part of that particle. Ziyu and Dingxue [17] introduced time-varying acceleration coefficients with exponential behavior (TACPSO) for providing the balance between two capabilities in the PSO algorithm: producing the new solutions and converging to global solution. Their work results indicate the relative success of the TACPSO algorithm in finding the global optima of the problems. Bao and Mao [18] suggested asymmetrical time-varying coefficients for improving the global search ability of the PSO algorithm and avoidance of the early convergence. It has been observed that this approach is superior to those methods that emphasize the use of symmetrical time-varying coefficients. Mirjalili et al. [19] proposed a new developed method of the PSO algorithm called autonomous groups particles swarm optimization (AGPSO). The main purpose of this method is to increase the diversity of solutions in the search space using the concept of autonomous groups inspired by the diversity of individuals in the natural colonies. To achieve this goal in the AGPSO method, three different types of the PSO algorithm called AGPSO1, AGPSO2 and AGPSO3 are developed. For each of these algorithms, four different strategies have been proposed to describe the dynamic behavior of the learning coefficients. In each of these strategies, the acceleration coefficients have incremental and decreasing behavior, and these behaviors are described using polynomial, exponential, and logarithmic functions. The results of this strategy show that the AGPSO method is superior than other optimization methods in terms of the overall search capability and the avoidance of premature convergence particularly for problems with high dimension. Ciu et al. [20] designed three different nonlinear settings to further investigate the potential advantages among two parameters; cognitive and social learning factors. Simulation results demonstrate that their proposed method performed in an effective manner especially for multimodel functions. Mahmoodabadi et al. [21] combined the PSO algorithm with new convergence and divergence operator (CDPSO) for producing a new particle, increasing the convergence of particles and avoiding of trapping in the local minima. They proved that this algorithm has very well performance in solving the complex optimization problems. Mortazavi et al. [22] introduced a new method that called integrated particle swarm optimization (iPSO). The iPSO method applies an improved Fly-back technique and the concept of the weighted particle for enhancing the standard PSO method.

In many studies to improve the optimization results, researchers have used the ability of the PSO in local search and the ability of various algorithms in global search and have presented different hybrid methods. For instance, PSO-ant colony optimization (ACO) [23], hybrid PSO and GA [24,25], PSO — gravitational search algorithm [26], hybrid PSO and dragonfly algorithm [27], PSO-multi crossover operator of GA and bee colony mechanism (HEPSO) [28].

To increase the diversity of the solutions obtained by PSO and solve the problem of early convergence and stuck in the local minimum, in some past researches, the authors combined the PSO algorithm with Levy flight. Levy flight uses the Levy distribution instead of uniform Gaussian distribution as a mechanism to generate the step sizes [29]. Levy flight, with small steps, enables the optimization algorithm to search around the current solutions. In addition, large jumps, which are occasionally produced by Levy flight, help to prevent the optimization algorithm from getting stuck in the local minimum. Recently, researchers have used Levy flight distribution to develop their algorithms, such as: Cuckoo search (CS) algorithm [7], dragonfly algorithm (DA) [30], Levy flight-based gray wolf optimization [31], Levy ant colony optimization algorithm [32] and multi-objective artificial bee colony algorithm with Levy flight [33], and etc.

Given that Levy flight strengthens the global search ability of original PSO algorithm. Levy flight can be used to increase the exploration capability of the PSO algorithm. Hence, in recent years, methods have been proposed based on the combination of PSO and Levy flight such as RPSOLF [34], LFPSO [35], and PSOLF [36].

In this study, the PSOSCALF algorithm is proposed for solving the disadvantages of the original PSO algorithm. This algorithm is based on combination of the PSO with the position updating equations of the SCA optimization method [37] and Levy flight. The SCA optimization algorithm is one of the population-based algorithms that recently has been proposed for solving the optimization problems. In this algorithm, each new solution is computed using the characteristics of sine and cosine functions and the absolute value of the distance between the current solution and the best member of the population that obtained so far. Oscillating behavior and varying the value of these trigonometric functions enable this algorithm for producing the exploration and exploitation capabilities. Therefore, the combination of Levy distribution and the SCA mechanism allows the design space to be searched further to find optimal solution. In the proposed PSOSCALF method, similar to [35], an index or limit value is considered for each particle. This index shows the number of iterations regarded the position of the particle which is not improved. If the position of the particle does not improve at the end of each iteration, this limit increases by 1 for the particle. If the calculated limit value exceeds a predetermined value, the new position of the particle is obtained by the new combination of SCA and Levy flight. Otherwise, the position of the particle will be updated by the PSO. The results show that the combination of SCA relationships and Levy flight with

PSO effectively overcomes the PSO's problems. Also, the proposed PSOSCALF method has been evaluated using the test functions with high dimensions and real constrained engineering problems. Also, the comparison with other PSO variants and other optimization methods has been done. The results show that in most cases, the proposed method has a much better performance.

The next sections of this article are as follows: In Section 2, fundamentals of the PSO optimization algorithm is described. In the third and fourth sections, the SCA optimization algorithm and the Levy flight are presented. The description of the details of the proposed PSOSCALF algorithm is given in Section 5. In Section 6, the ability of the proposed algorithm to optimize the benchmark functions is investigated and compared with the PSO family algorithms and other algorithms. The ability of the PSOSCALF method to solve the engineering problems in Section 7 has been evaluated. Finally, this paper is concluded in Section 8.

2. Particle swarm optimization

The fundamental principles of the particle swarm optimization algorithm have been derived from studying the social life of the fishes and birds that live in groups [8]. In this technique, each particle is a solution to the desired optimization problem and moves using its the velocity and position vectors in the design space to find a new solution. In this study, the vectors $\vec{X}_i = \{X_{i1}, X_{i2}, \ldots, X_{id}\}$ and $\vec{V}_i = \{V_{i1}, V_{i2}, \ldots, V_{id}\}$ indicate the velocity and position of each particle in the search space for a d-dimensional problem, respectively.

In the early stages of implementing this algorithm, firstly, the position and velocity vectors of all particles are generated randomly. Then, each particle moves in the design space using the best position experienced by that particle ($\vec{X}_{pBest}$) and the best solution obtained by all particles ($\vec{X}_{gBest}$). The updating equations of the velocity and position for each particle are as follows: [38]:

$$\vec{V}_i(t+1) = w\vec{V}_i(t) + c_1 r_1 \left(\vec{X}_{pBest_i} - \vec{X}_i(t)\right) + c_2 r_2 \left(\vec{X}_{gBest} - \vec{X}_i(t)\right) \quad (1)$$

$$\vec{X}_i(t+1) = \vec{X}_i(t) + \vec{V}_i(t+1) \quad (2)$$

where $\vec{V}_i(t+1)$ and $\vec{X}_i(t+1)$ are velocity and position vectors of particle i at iterations $t+1$ respectively. c_1 and c_2 are personal learning factor and social learning factor respectively. r_1 and r_2 are random number in the interval [0,1].

As seen in Eq. (1), the motion of each particle in the design space is affected by three terms: the previous velocity vector, the second part of the velocity vector that named "cognitive" component and directs the particle towards the best solution experienced by the particle itself in the search space, and the last part that named "social" component and leads to move the particle towards the best solution obtained by all particles [36].

Shi and Eberhat [38] added the inertia weight parameter to the velocity updating equation in the PSO algorithm for investigating the effect of the previous velocity on the movement of a particle. Large values of the inertial weight lead to a global search and small value of this factor makes local search [39]. Therefore, by changing the value of w, the search ability is dynamically adjustable. In this paper, the adaptive coefficient of the inertia weight proposed in [40] is used:

$$w(t) = w_f + \left(\frac{1 + \cos\left(\frac{\pi t}{t_{\max}}\right)}{2}\right)^k (w_i - w_f) \quad (3)$$

w_i and w_f are the initial and final values of the weight factor respectively. $t_{\max}$ is the maximum number of repetition, t is the current iteration number and k is a constant number that indicates the decreasing intensity of the weight factor.

During the optimization process, if c_1 and c_2 treat as the incremental and decreasing monotonic functions respectively, then, it leads to exploration capability at the beginning and exploitation capability in the final stages. In this paper, the following equations [16] in which c_1 linearly decreases and c_2 increases linearly have been used:

$$c_1(t) = c_{1\min} + \left(\frac{t_{\max} - t}{t_{\max}}\right)(c_{1\max} - c_{1\min}) \quad (4)$$

$$c_2(t) = c_{2\max} + \left(\frac{t_{\max} - t}{t_{\max}}\right)(c_{2\min} - c_{2\max}) \quad (5)$$

where $c_{1\min}$ and $c_{2\min}$ are the minimum values of personal and social learning factors respectively, and $c_{1\max}$ and $c_{2\max}$ are the maximum values of personal and social learning factors respectively.

3. Sine Cosine algorithm (SCA)

In 2016, Mirjalili proposed a new population-based optimization algorithm called the Sine Cosine algorithm (SCA) in [37] to solve the optimization problems. SCA produces several random solutions, and then approaches them to the optimal solution by means of equations that include sine and cosine functions. Of course, in these relationships, in addition to the features of these functions, the distance between the best member of the population and each of the solutions affects their movement. The SCA technique, which will be further described in its relationship, is capable of creating a balance between exploitation and exploration by employing far fewer operators than other algorithms. This approach applies the following relationships to update the solutions for both exploration and exploitation phases [37]:

$$\vec{X}_i(t+1) = \vec{X}_i(t) + r_1 \sin(r_2) \left| r_3 P^t - \vec{X}_i(t) \right| \quad (6)$$

$$\vec{X}_i(t+1) = \vec{X}_i(t) + r_1 \cos(r_2) \left| r_3 P^t - \vec{X}_i(t) \right| \quad (7)$$

In above equations, $\vec{X}_i(t)$ and $\vec{X}_i(t+1)$ are the positions of current solution in the iterations t and $t+1$ respectively. Parameters r_1, r_2 and r_3 are random numbers and P^t is the position of the best member of the population or destination point in the iteration t and $||$ is absolute value. By combining these two equations, position of the solution updates as follows [37]:

$$\vec{X}_i(t+1) = \begin{cases} \vec{X}_i(t) + r_1 \sin(r_2) \left| r_3 P^t - \vec{X}_i(t) \right|, r_4 < 0.5 & \text{(a)} \\ \vec{X}_i(t) + r_1 \cos(r_2) \left| r_3 P^t - \vec{X}_i(t) \right|, r_4 \geq 0.5 & \text{(b)} \end{cases} \quad (8)$$

where $r_4 \in [0, 1]$ is a random value and operates as the switching factor between Eqs. (8)a and (8)b. The parameters r_1, r_2, r_3 and r_4 are the four effective parameters in the SCA algorithm for determining the position of the new solution. The r_1 parameter specifies the next position of the solution that can be located in space between the current solution and the destination point ($r_1 < 1$, exploitation) or outside it ($r_1 > 1$, exploration). In [37], the following equation is proposed for the calculation of r_1:

$$r_1 = a\left(1 - \frac{t}{t_{\max}}\right) \quad (9)$$

where t and $t_{\max}$ are the current iteration and the maximum iteration respectively, and a is constant number. In this article similar to [37], $a = 2$ is considered.

The parameter r_2, which is a random number in the interval $[0, 2\pi]$, determines how far the solution can be towards or outwards the destination point. $r_3 \in [0, 2]$ assigns a random weight to the

destination point to determine the effect of this point on the distance definition [37].

The periodic behavior of the sine and cosine trigonometric functions allows a solution to be generated around the other. This feature ensures the exploitation capability. On the other hand, if a new solution is generated outside of the space between the current solution and the best solution that obtained so far, then global search capability is generated during optimization. In the SAC algorithm, this ability is available by varying the value of sine and cosine functions [37].

4. Levy flight

Recently, researchers have utilized Levy flight in different optimization techniques for producing the random step size in the design region [3,34–36]. Levy flight is one of the random processes with non-Gaussian distribution. This distribution is often in terms as $L(s) \sim |s|^{-1-\beta}$ where in this formula, β parameter is an index in the interval $(0, 2)$. From a mathematical point of view, a simple form of mathematical formula for Levy flight distribution can be written as follows [3]:

$$L(s,\gamma,\mu) = \begin{cases} \sqrt{\frac{\gamma}{2\pi}} \exp\left(-\frac{\gamma}{2(s-\mu)}\right) \frac{1}{(s-\mu)^{\frac{3}{2}}}, & 0 < \mu < \infty \\ 0, & otherwise \end{cases} \quad (10)$$

In this distribution, μ and s are transmission parameter and samples, respectively. Parameter γ is a parameter that controls scale of Levy flight distribution. Generally, the definition of Levy flight distribution in the Fourier transform form is expressed as follows [3]:

$$F(k) = \exp\left(-\alpha |k|^{\beta}\right), \quad \beta \in (0, 2] \quad (11)$$

where α is a scale parameter. The analytical form of this integral is can be calculated for a few exceptional cases of β. The method of calculating the step length s is discussed in [2] in detail. In the following, it will be discussed briefly. The step length s can be determined based on the Mantegna's algorithm as follows [34–36]:

$$s = \frac{u}{|v|^{\frac{1}{\beta}}} \quad (12)$$

where u and v parameters in the above equation have Gaussian distribution and are obtained as follows:

$$u \sim N\left(0, \sigma_u^2\right), \quad v \sim N\left(0, \sigma_v^2\right) \quad (13)$$

$$\sigma_u = \left\{ \frac{\Gamma(1+\beta)\sin(\frac{\pi\beta}{2})}{\Gamma\left[\frac{(1+\beta)}{2}\right]\beta\, 2^{\frac{(\beta-1)}{2}}} \right\}^{\frac{1}{\beta}}, \quad \sigma_v = 1 \quad (14)$$

Then, step size is calculated by

$$stepsize = scale \times s \quad (15)$$

Here, *stepsize* is the step size in the search space, and the factor s is dependent on the dimension of the desired problem. Otherwise, Levy flight may have highly aggressive behavior and leads to generate new solutions outside of the design space [34]. As will be explained in the next section, *stepsize* value will be added to the updating equations of the SCA algorithm for finding the position of the new particle. In other words, the Levy flight distribution is effective mathematical operator for producing the varied solutions in the design space and increasing the exploration capability of the PSO algorithm.

5. The proposed PSOSCALF algorithm

Although the PSO algorithm has the excellences of the high speed computation, the small number of parameters and the simplicity in implementation; it has two major deficiency as premature convergence and falling into the local minima. In recent studies such as [34,35], researchers introduced various versions of the PSO algorithm by merging the velocity and position updating equations in the PSO with Levy flight. As, the use of Levy flight leads to a more effective search in the search space with large jumps, various solutions are produced.

As described in Section 3, sine and cosine trigonometric functions and the parameters used in the SCA algorithm have features that provide exploration and exploitation capabilities for this algorithm. Here, In order to produce more varied solutions throughout the design space and increase the exploration capability of the SCA algorithm, Levy flight distribution is added to the SCA equations as follows:

$$\vec{X}_i(t+1) = \begin{cases} Levy_{walk}(\vec{X}_i(t)) + r_1 \sin(r_2) \left| r_3 \vec{X}_{gBest} - \vec{X}_i(t) \right|, r_4 < 0.5 & (a) \\ Levy_{walk}(\vec{X}_i(t)) + r_1 \cos(r_2) \left| r_3 \vec{X}_{gBest} - \vec{X}_i(t) \right|, r_4 \geq 0.5 & (b) \end{cases} \quad (16)$$

In above equations, expression $Levy_{walk}(\vec{X}_i(t))$ is added to the SCA relationships, and is calculated in the same way as the PSOLF method [35]:

$$Levy_{walk}(\vec{X}_i(t)) = \vec{X}_i(t) + \overrightarrow{step} \oplus \overrightarrow{random}(size(\vec{X}_i(t))) \quad (17)$$

$$\overrightarrow{step} = stepsize \oplus \vec{X}_i(t)$$

Dimension of vector $\overrightarrow{random}(size(\vec{X}_i(t)))$ is equal to dimension of the desired optimization problem. *stepsize* is calculated by Eq. (15). $\vec{X}_{gBest}$ is global optimum obtained so far and $\oplus$ is element-wise multiplication.

In this article, the new hybrid equations proposed in Eqs. (16) and (17) are added to the PSO algorithm until all of particles to be able to better search the design space for detecting the global optima. In the new proposed algorithm, an index named "limit value" is assigned to each particle, similar to LFPSO [35]. This index for each particle is equal to the number of iterations that the position of this particle is not improved. In other words, if the position of this particle does not get better at the end of each repetition in the design space, limit value for this particle is increased by one unit. Eventually, if the calculated limit value exceeds a predetermined value, a new position for this particle will be created by the combination of SCA and Levy flight.

The pseudo code related with the PSOSCALF algorithm is shown in Fig. 1. According to this figure, *npop* is population size, *MaxIt* is maximum iteration, $X_{\min}$ and $X_{\max}$ are the lower bound and upper bound in search space and *dim* is the dimension of the problems. These parameters are set at the beginning of the proposed algorithm. As shown in Fig. 1, the position and velocity of particles are initially generated randomly. Then, by calculating the value of the objective function, $\vec{X}_{pBest_i}$ is determined for each particle and $\vec{X}_{gBest}$ for the entire population. In each iteration, the value of the parameters c_1, c_2 and w are calculated utilizing Eq. (3) to (5). Before updating the vector of the velocity of each particle, the limit value of that particle is checked. If it is smaller than the predetermined value, the velocity and position of the particle are updated by Eqs. (1) and (2) in the simple PSO. Otherwise, the next position of that particle is determined by Eqs. (16) and (17), which are a combination of the SCA algorithm and Levy flight. In this case, a random value is assigned to that particle. If rand ( ) <0.5, the particle

- Initialize the parameters (*npop, MaxIt,* X_{min}, X_{max}, *dim*)
- trial (keeps the limit value for each particle) = 0
- initialize the particles with random positions (X_i) and random velocity (V_i) within initialization range in the problem space
- Evaluate the fitness
- Set X_i to be X_{pBest}
- Set the particle with best fitness to be X_{gBest}

for t = 1:*MaxIt*
  Calculation of $c_1(t), c_2(t)$ and $w(t)$
  for i = 1:*npop*
    if trial(i)<limit
      Update the velocity V_i of particle using Eq. (1)
      Update the position X_i of particle using Eq. (2)
    else
      if rand() < 0.5
        Update the position X_i of particle using Eq. (16)
      else
        Update the position X_i of particle using Eq. (17)
      end if
    end if
    Position's value is brought to the boundary value when its values are moved out of the boundary in search space
    Evaluate fitness value for new particle X_i
    if f($X_i(t)$) < f(X_{pBest})
      trial(i) = 0;
      $X_{pBest} = X_i$
    else
      trial(i) = trial(i)+1
    end if
    if f(X_{pBest}) < f(X_{gBest})
      $X_{gBes} = X_{pBest}$
    end if
  end for
end for

Fig. 1. Pseudo code of the PSOSCALF algorithm.

position is computed by Eq. (15), and if rand ()> 0.5, the position of the particle is calculated by Eq. (16). The new obtained position should be in the range ($X_{\min}$, $X_{\max}$). If it exceeds this range, it will be taken by the following operators to the range ($X_{\min}$, $X_{\max}$):

$$X = \min(X, X_{\max})$$
$$X = \max(X, X_{\min}) \quad (18)$$

With updating the velocity and position of all particles, new particles are produced in the design space. In the next step, objective function is determined for all new particles. Then, objective value of each new particle is compared with its personal best experience ($\vec{X}_{pBest_i}$). If the position of this particle is improved, then $\vec{X}_{pBest_i}$ is updated and its index is reset to zero and otherwise this index increases by one unit. Finally, $\vec{X}_{gBest}$ is determined for the whole particles. The above process is repeated until the iteration number is equal to *MaxIt*. The flowchart of the PSOSCALF algorithm is presented in Fig. 2.

6. Result and discussion

Using benchmark functions with different features is a common method for evaluating optimization algorithms with stochastic nature. By using these functions, it can be ensured that the results of an algorithm are not accidental. So far, researchers have used several benchmark functions with different characteristics to evaluate their algorithms. In this section, we first introduce the test functions used in this work and then, to evaluate the PSOSCALF algorithm, the results of this algorithm are compared with the results of the various versions of the PSO algorithm (Section 6.3) and the other algorithms (Section 6.4).

Table 1
Parameter settings of PSOSCALF.

Parameter	Value	Parameter	Value
c_{1min}	0.5	w_{min}	0.4
c_{2min}	0.5	w_{max}	0.9
c_{1max}	2.5	K	10
c_{2max}	2.5	*npop*	50
limit	10	*MaxIt*	500
β	1.5	*scale*	0.01

6.1. Parameter settings for PSOSCALF method

The control parameters that affect the performance of the proposed PSOSCALF method are adjusted according to Table 1. In this paper similar to [34,36], Levy flight parameters β and *scale* are considered 1.5 and 0.01 respectively.

6.2. Benchmark functions

In this study, the 23 benchmark functions presented in Tables 2–4 are used to evaluate the proposed PSOSCALF algorithm. Mathematical formula of these test functions are available in Tables 2–4. These functions are classified into three categories: unimodal function, multimodal function, fixed-dimension multimodal.

The unimodal test functions listed in Table 2 due to having only one optimal point, are an ideal option for testing the convergence rate and the exploitation of optimization technique. In contrast, function of the second and third groups presented in Tables 3 and 4, have different local minima solutions in addition to the

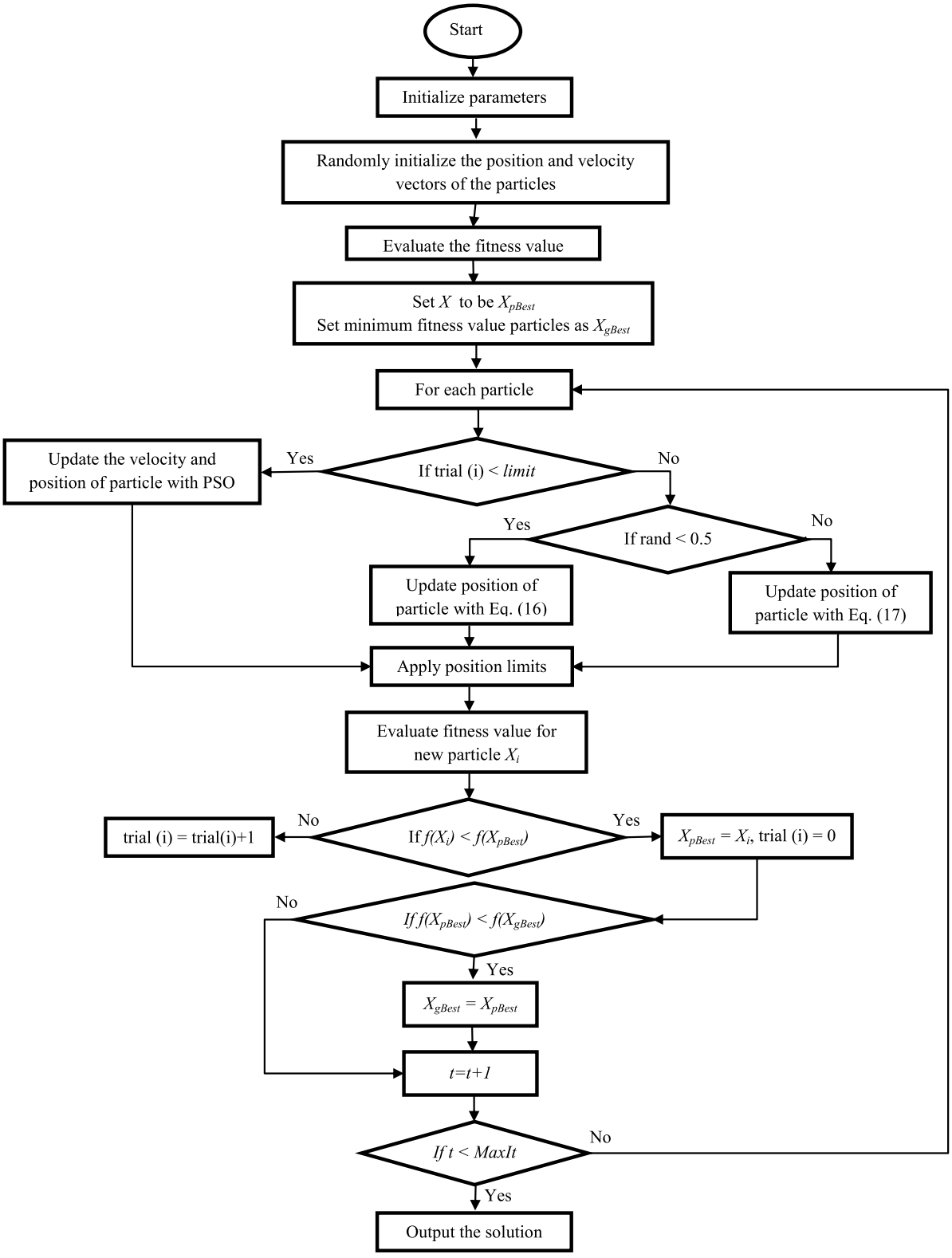

Fig. 2. Flowchart of the proposed PSOSCALF algorithm.

global optima. These functions are useful for evaluating exploration algorithms. Finally, according to the mathematical formula of the fixed-dimension multimodal functions given in Table 4, there are two points about these functions: first, the number of the design variables of these function cannot be changed, and second, the search space for these functions is different with multimodal functions [6]. The mathematical characteristics of the three above-mentioned classes consist of the mathematical formula, the minimum value, the number of design variables, and the range of the search which are given in Tables 2–4.

Table 2
Unimodal benchmark functions [6].

Function	*dim*	$[X_{min},X_{max}]$	f_{min}
$F_1(x) = \sum_{i=1}^{n} x_i^2$	30	[−100,100]	0
$F_2(x) = \sum_{i=1}^{n} \lvert x_i^2 \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	[−10,10]	0
$F_3(x) = \sum_{i=1}^{n} \left(\sum_{j-1}^{i} x_j \right)^2$	30	[−100,100]	0
$F_4(x) = max_i\{\lvert x_i \rvert, 1 \le i \le n$	30	[−100,100]	0
$F_5(x) = \sum_{i}^{n-1} [100 \left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2]$	30	[−30,30]	0
$F_6(x) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	30	[−100,100]	0
$F_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0, 1)$	30	[−1.28,1.28]	0

6.3. Comparison of PSOSCALF and PSO families

Here, to evaluate the PSOSCALF algorithm, the results of this approach are compared with various versions of the PSO algorithm. For this purpose, the examination functions presented in Tables 2–4 are used. The PSO family algorithms that their results are reported in this study, are: PSO [8], HPSO-TAVC [16], TACPSO [17], MPSO [18], AGPSO [19], HEPSO [28], RPSOLF [34] and LFPSO [35]. Since meta-heuristic algorithms have stochastic nature, their results might be unreliable for just one run. Therefore, for the comparison of all algorithms with each other, the results for 30 independent runs are presented in Appendix A for benchmark functions introduced. In each run, the initial population is randomly assigned. To compare these algorithms with each other in finding the optimum solution of a specific benchmark function, the results from different runs are reported as statistical characteristics, mean, standard deviation and algorithm rank in Tables A.1–A.3. The ranking of these algorithms is based on their competency in statistical results, namely, the mean and standard deviation which are presented in Tables A.1–A.3. In these tables, the best mean and best standard deviation and the algorithm rank for each benchmark function are shown in bold.

6.3.1. Investigation of exploitation and exploration capabilities

In this article, by examining the results of minimizing the unimodal functions, the ability of various algorithms to exploit the solutions in the search space has been evaluated. The results for these functions are presented in Table A.1. As shown in this table, among these methods, only PSOSCALF and RPSOLF methods are ranked 1 for some benchmark functions. The PSOSCALF method is ranked 1 in the F5, F6 and F7 and the RPSOLF method for the four F1, F2, F3, and F4 functions. But PSOSCALF is ranked second in the rest of the functions, while RPSOLF for the F6 is ranked eighth. The overall ranking of the PSOSCALF algorithm and other PSO families are presented in Table A.1. As can be seen, the PSOSCALF algorithm is more effective than other algorithms used in this work in finding the global optima of the unimodal functions (F1–F7), and is ranked 1. On the other hand, standard deviation indicates the stability of an algorithm in various runs. From the results of the standard deviation presented in Table A.1, it is concluded that the proposed method has a good stability and is more robustness than the rest.

Functions F8 to F23 are multimodal and are suitable for evaluation of exploration capability of algorithms due to having several local optima. Results for F8 to F13 functions are presented in Table A.2. As can be seen, PSOSCALF, with the exception of F10, has the best results. Of course, the proposed method in the F10 also ranked second. Therefore, the preference of the proposed method in this paper is clearly recognizable. The RPSOLF method has the first rank in F9, F10, and F11, and in both cases F9 and F11 are similar to those proposed in this paper. Nevertheless, RPSOLF ranked 8 in F8 and 7 in F12 and 9 in F13, ranked fourth among all methods. The standard deviation values for all functions imply that the PSOSCALF method is stable for various runs.

Results for functions (F14–F23) are presented in Table A.3. As can be seen, the HPSO-TAVC method is not first rank in any of the functions. The proposed PSOSCALF method is first rank for the five functions F16, F18, F21, F22 and F23, and is third rank for F14 and F17 and is fifth rank for F19 and F20. On the other hand, in the cases

Table 3
Multimodal benchmark functions [6].

Function	*dim*	$[X_{min},X_{max}]$	f_{min}
$F_8(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	30	[−500,500]	−418.9829 × dim
$F_9(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	30	[−5.12,5.12]	0
$F_{10}(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{2}\sum_{i=1}^{2}\cos(2\pi x_i)) + 20 + e$	30	[−32,32]	0
$F_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	[−600,600]	0
$F_{12}(x) = \frac{\pi}{n}\left\{10\sin(\pi y_1) + \sum_{i=1}^{n}(y_i - 1)^2\left[1 + 10sin^2(\pi y_i)\right] + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i + 1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, x_i > a \\ 0, -a < x_i < a \\ k(-x_i - a)^m, x_i < -a \end{cases}$	30	[−50,50]	0
$F_{13} = 0.1\left\{sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2\left[1 + sin^2(3\pi x_1 + 1)\right] + (x_n - 1)^2\left[1 + sin^2(2\pi x_n)\right]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	30	[−50,50]	0

Table 4
Fixed-Dimension multimodal benchmark functions [6].

Function	*dim*	$[X_{min},X_{max}]$	f_{min}
$F_{14}(x)=(\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6})^{-1}$	2	[−65,65]	1
$F_{15}(x)=\sum_{i=1}^{11}[a_i-\frac{x_1(b_i^2+b_1x_2)}{b_i^2+b_1x_3+x_4}]^2$	4	[−5,5]	0.00030
$F_{16}(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	2	[−5,5]	−1.0316
$F_{17}(x)=(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6)^2+10\left(1-\frac{1}{8\pi}\right)cosx_1+10$	2	[−5,5]	0.398
$F_{18}(x)=\left[1+(x_1+x_2+1)^2\left(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2\right)\right]\times[30+(2x_1-3x_2)^2\times(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)$	2	[−2,2]	3
$F_{19}(x)=-\sum_{i=1}^{4}c_i\exp(-\sum_{j=1}^{3}a_{ij}(x_j-p_{ij})^2)$	3	[0,1]	−3.86
$F_{20}(x)=-\sum_{i=1}^{4}c_i\exp(-\sum_{j=1}^{6}a_{ij}(x_j-p_{ij})^2)$	6	[0,1]	−3.32
$F_{21}(x)=-\sum_{i=1}^{5}[(X-a_i)(X-a_i)^T+c_i]^{-1}$	4	[0,10]	−10.1532
$F_{22}(x)=-\sum_{i=1}^{7}[(X-a_i)(X-a_i)^T+c_i]^{-1}$	4	[0,10]	−10.4028
$F_{23}(x)=-\sum_{i=1}^{7}[(X-a_i)(X-a_i)^T+c_i]^{-1}$	4	[0,10]	−10.5363

where the PSOSCALF method is not ranked 1, the results of most methods are very close. All of these points indicate the superiority of the proposed algorithm to the other methods discussed in this paper. The second and third ranks belong to the HEPSO and MPSO algorithms, respectively.

6.3.2. Analysis of convergence behavior

If at the beginning of the optimization process, the particles explore whole the design area for producing the various solutions and then gradually converge to near the best solution, this behavior will guarantee the ability of the algorithm to find the global optimization. To investigate the behavior of the PSOSCALF technique in all optimization stages, the fluctuations of the first dimension in the first particle are provided in Fig. 3. These curves are obtained for functions F1, F3, F7, F10, F13 and F18. As it is seen, at the beginning of all the curves, there are severe fluctuations and then these fluctuations are reduced. The fact is that this particle initially explores the design space and then converges around the best solution.

On the other hand, it is obvious that by studying the behavior of the first particle and observing its recovery, it cannot be concluded that the positions of all particles in all stages of optimization are improving. Therefore, in order to investigate the behavior of all the particles during the optimization process, the average fitness of all solutions is shown in Fig. 4. Results are obtained for F1, F3, F7, F10, F13 and F18 functions. As can be seen, for all functions exception of F18, in the exploration phase, the high fluctuations are occurred and then gradually are decreased, and finally, in the exploitation phase, the value of the average fitness has very few changes. But this is not true for F18. Maybe at first glance, by observing the fluctuations of the F18 function in Fig. 4, it is concluded that the PSOSCALF methods is inefficient in determining the global optima of this function. But according to the results of the function F18 in Table A.3, it is clearly seen that the PSOSCALF algorithm is able to determine the global optima for this function. To justify these two observations, we can point to the presence of particles that try to move in the search space during the optimization process to find the global optimum.

So far, particles behavior during optimization in the PSOSCALF algorithm has been investigated. In order to investigate the convergence rate for five methods PSOSCALF, RPSOLF, LFPSO, PSO and MPSO, the convergence behaviors for all three categories of benchmark functions are indicated in Figs. 5–7. The authors have been rewritten the source code of each algorithm and for all the algorithms, the maximum number of function evaluations (FEs) is used as stopping criterion. In Figs. 5–7, the average of the best solutions in 30 runs with 200 000 FEs is plotted for all the test benchmark functions.

In Fig. 5, the convergence results are plotted for unimodal functions. As shown in this figure, RPSOLF method procures global optimum 0 for three functions F1, F3 and F4 and attains near optimal solution for F2. On the other hand, other methods improve the solutions for these functions continuously and PSOSCALF method provides better results. Convergence curve for F6 shows that all methods do not improve solution and stuck in local optimum. Given the curves for functions F5 and F7, the convergence behavior of the methods used in this paper is similar to together, with the difference that the PSOSCALF method improves solutions to a less value. The curves for the F5, F6 and F7 functions indicate the exploitation capability for the proposed method in the final stages of optimization.

In Figs. 6 and 7, the convergence behavior of different methods for functions F8 to F23 is presented. Fig. 6 show that the RPSOLF method converges and acquires the global optimum of 0 for F8, F9 and F11 at 11 300, 43 600 and 48 950 FEs respectively. Also, the proposed PSOSCALF method reaches optimum solution 0 for F9 and F11 at 2350 and 5750 FEs respectively. Convergence curves in Fig. 6 show that all of the methods get stuck in local minima for F10, F12 and F13 functions. Given the convergence behavior of function F10, it can be seen that the PSOSCALF algorithm achieves better result than other methods at 63 550 FEs. By investigating F12 function, it is observed that the PSOSCALF and LFPSO methods

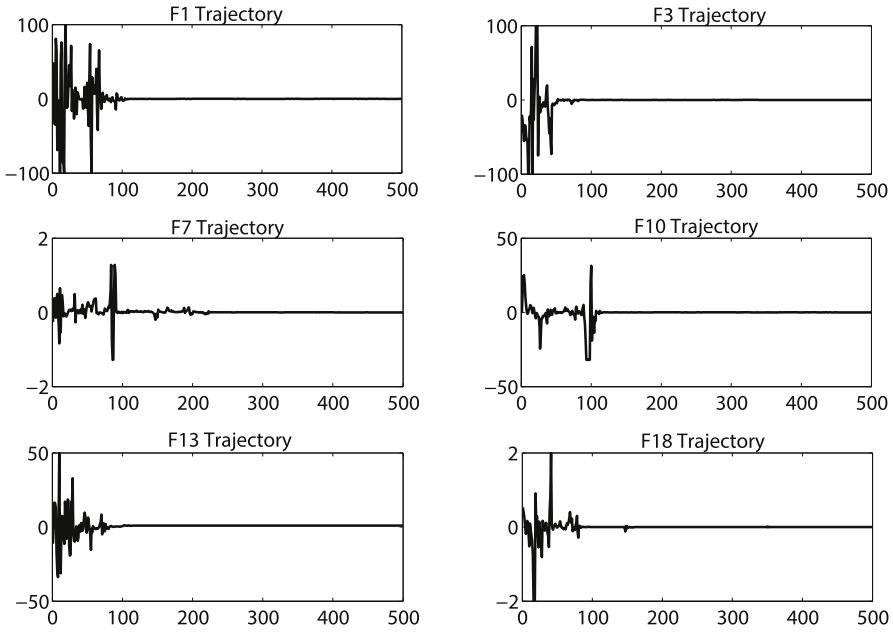

Fig. 3. Trajectory of the first dimension in the first particle when solving the benchmark functions.

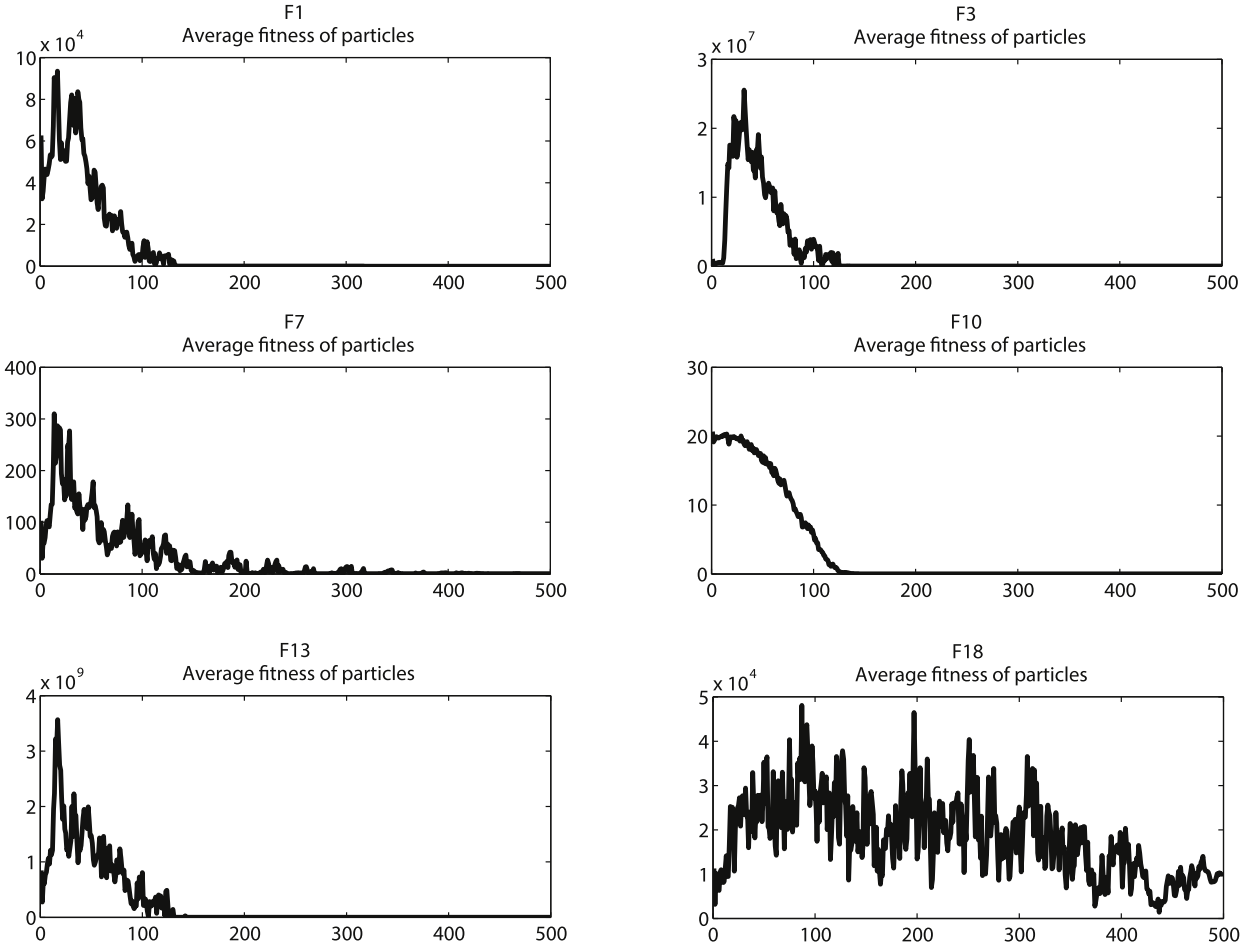

Fig. 4. Average fitness of all particles during the optimization process.

improve solution in finding the optimal solution of this function and converge to near optimal solution at 97 600 and 182 400 FEs respectively.

By preliminary review of Fig. 7, it can be seen that the convergence behavior of the methods used in this paper for fixed-dimension multimodal functions is very similar to each other. By a

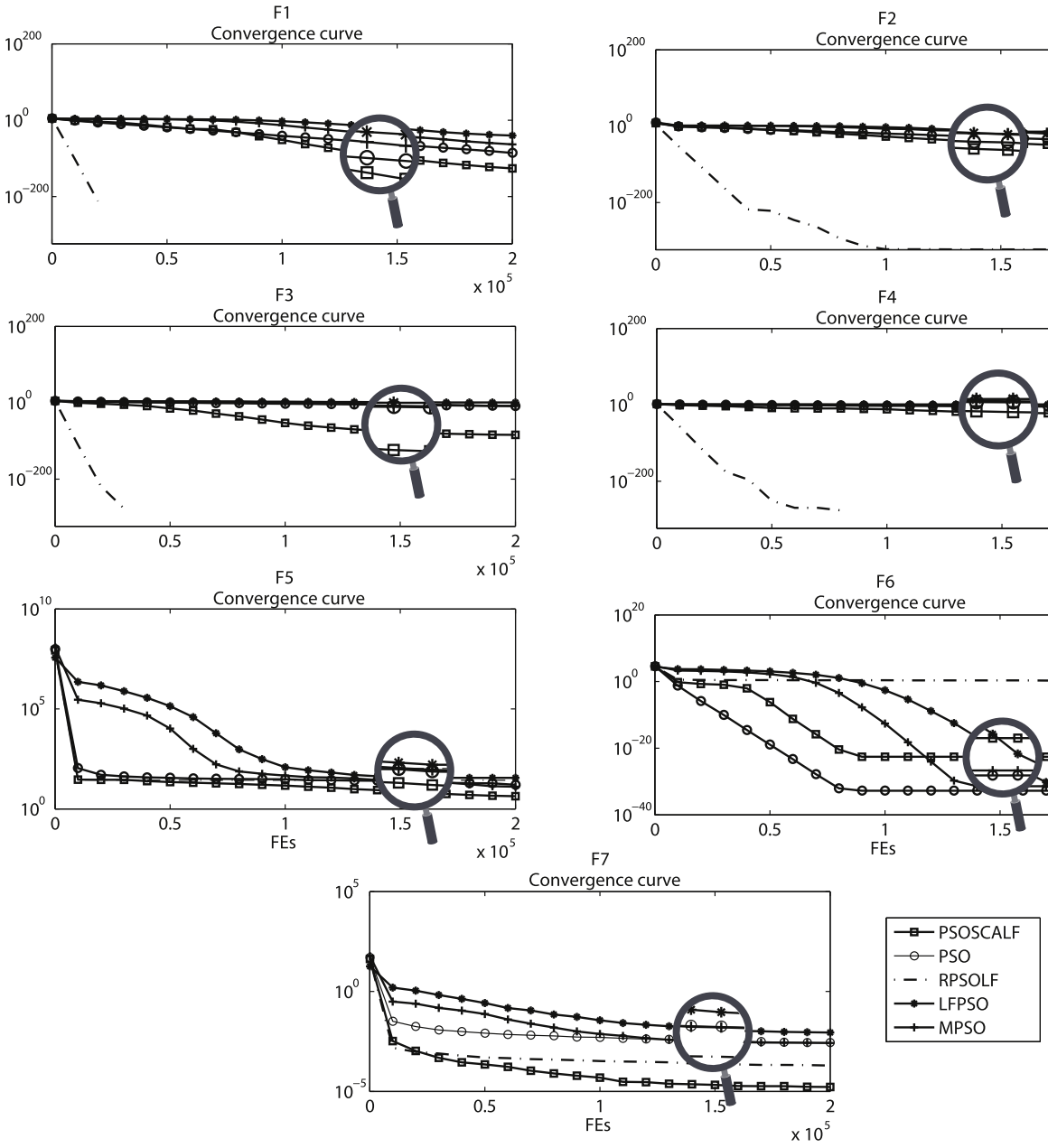

Fig. 5. Convergence curve for unimodal functions.

closer investigation of the convergence curves, it can be seen that the PSOSCALF method obtains global optimum for F14 to F23 at 14 100, 59 400, 12 050, 19 800, 15 650, 8650, 26 500, 36 500, 36 300, 33 200 and 39 000 FEs respectively. The LFPSO method acquires optimum results for F14 and F16 to F23 at 6400, 27 150, 25 400, 305 500, 22 550, 37 100, 71 700 and 36 150 FEs respectively. On the other hand, this method does not carry out well for function F15 and gets stuck in local minimum. According to above results, the proposed method has better convergence rate than the LFPSO method in seven functions among all functions F14–F23. Another point that can be seen in Fig. 7 is that the MPSO and PSO methods are able to obtain optimum value for only four functions F16 to F19. These methods get stuck in local minimum for rest of examination functions.

Above analysis and Figs. 5–7 show that the methods PSOSCALF, RPSOLF, LFPSO, MPSO and PSO are able to achieve global optimum at thirteen, nine, seven, six and four functions respectively. It is seen from the investigation of the convergence behavior of the benchmark functions that in cases where the proposed method is able to determine the optimal solution, this method has a good convergence rate and achieves optimal results quickly. Finally, the results and above explanations prove that the new hybrid algorithm presented in this article is able to determine the global minima or near optimal solution in most benchmark functions and has superiority over different methods.

6.4. Comparison of PSOSCALF algorithm and other Meta-Heuristic algorithms

In this section, the results of the proposed PSOSCALF method are compared with meta-heuristic algorithms. Some of these algorithms have been developed by researchers recently. The methods used here are: GA, WOA [6], DA [30], SCA [37], ALO [41], FA [42], MFO [43] and MVO [44]. The results for the functions presented in

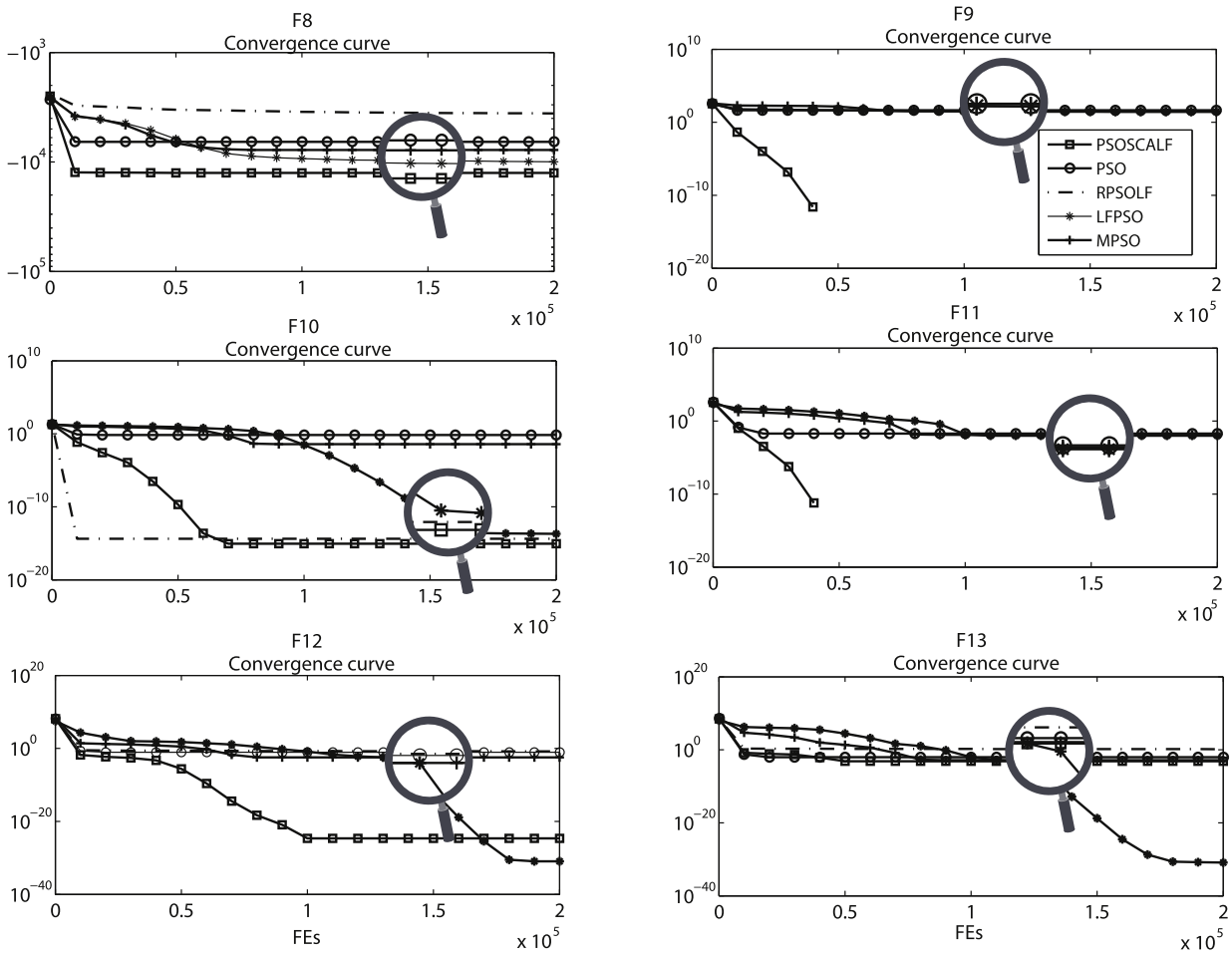

Fig. 6. Convergence curve for multimodal functions.

Tables 2–4 are calculated. The parameters of the PSOSCALF algorithm are the same as the parameters in Table 1. For all algorithms, the maximum iteration is $MaxIt = 500$, the population size is $npop = 50$. The above algorithms are composed of operators that produce random solutions in the design space. Therefore, in this paper, in order to have accurate interpretation of the performance of these algorithms, these methods to be run 30 times for optimizing each benchmark function and the results are reported in Appendix B in Tables B.1–B.3 as stochastic terms: mean, standard deviation and algorithm rank. Finally, the best solution of the benchmark functions is specified in bold.

According to Table B.1, the WOA algorithm has the best results for both F1 and F2 and the proposed PSOSCALF algorithm is ranked second for these two functions. But, the results of the optimization of F3 to F7 functions indicate the superiority of the PSOSCALF algorithm compared to other methods. Therefore, by investigating the results presented in Table B.1 and observing the final rank of these algorithms, it is concluded that the PSOSCALF method is more efficacious than other methods in finding the global optima of unimodal functions.

Table B.2 presents results for multimodal functions. As it can be seen, the proposed method is comparable to WOA for finding the global optimum of F9 and F11 functions. Also, the PSOSCALF method finds the best solution for the rest of the multimodal functions except F10. In Table B.3, the results are presented for fixed-dimension multimodal functions. As seen in this table, the PSOSCALF method performs better than other algorithms in achieving the minimum of five functions F15, F16, F21, F22 and F23. For example, the optimal values obtained by other methods for F21, F22 and F23 functions are far from the actual optimal of these functions.

According to the results of Tables B.1–B.3, it can be observed from the final rank of all algorithms that the proposed PSOSCALF method is an extremely effective and efficient algorithm for optimizing multimodal and unimodal functions.

6.5. Investigation of PSOSCALF' performance using Wilcoxon rank sum test

The statistical parameters such as mean and standard deviation are appropriate for general investigation of the ability of the optimization algorithms. In this section, in order to determine that the obtained results in previous sections are not stochastic, different non-parametric statistical tests such as Wilcoxon's test can be used. In here, Wilcoxon's rank sum test is selected for determining whether there is a statistical significance difference between the results of the PSOSCALF technique and other optimization algorithms. This non-parametric test is implemented for the results of 30 independent runs at significance level $\alpha = 0.05$. The calculated results are reported in Tables C.1 and C.2 in terms of *P-value*, *h-value* and *z-value* for benchmark functions F1–F23. The results of Wilcoxon's test for comparison of PSOSCALF with other PSO variants and other algorithms are presented in Tables C.1 and C.2, respectively. The term of *N/A* means that both algorithms are successful in determining optimal point of a specific function in all the runs and statistical Wilcoxon test is not applicable.

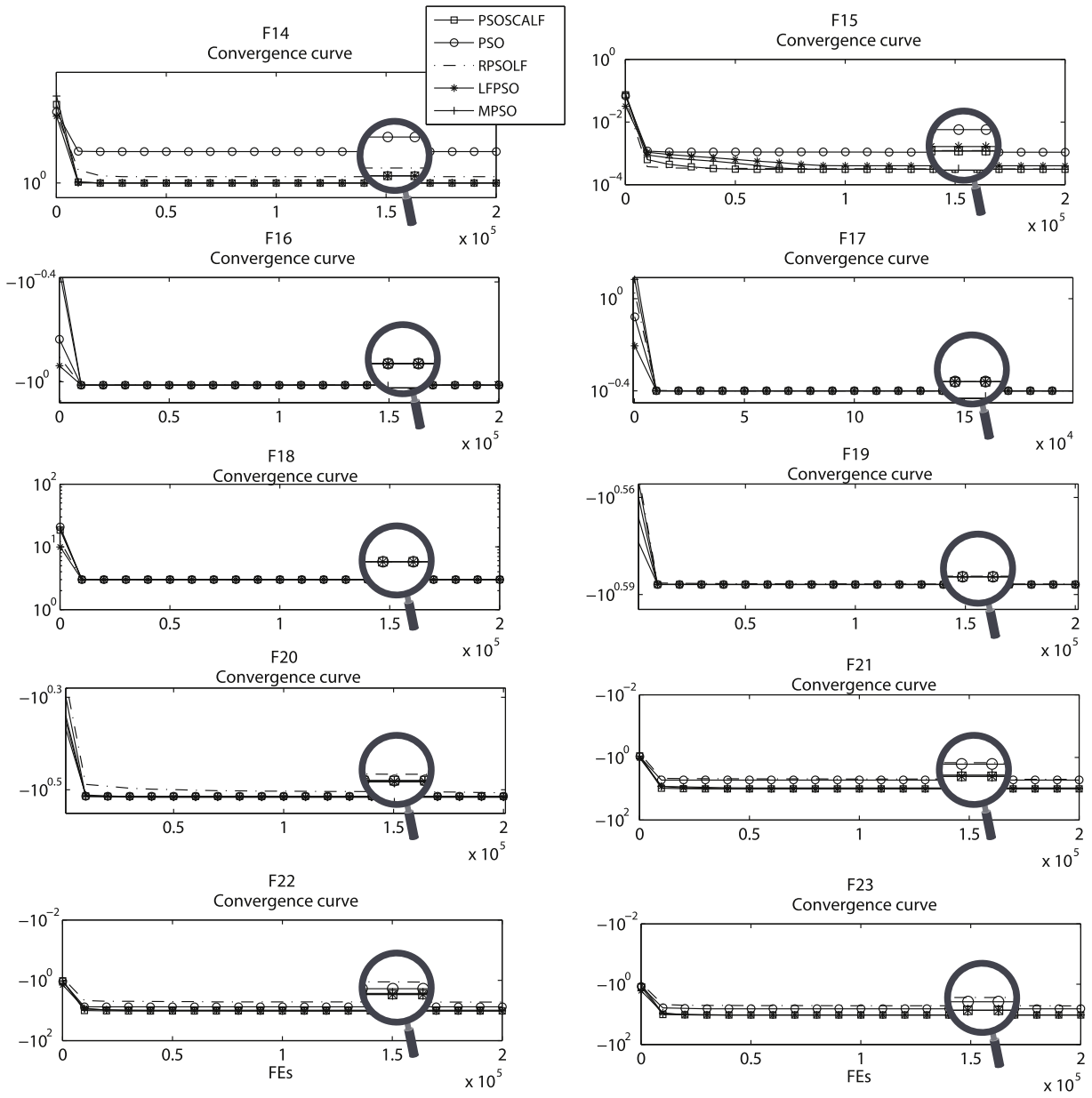

Fig. 7. Convergence curve for fixed-dimension multimodal functions.

By investigating the results in Tables C.1 and C.2, The following observations can be found:

- The MFO method and all of the PSO family algorithms except HEPSO are successful in finding the optimal point of function F17 in all the runs and the statistical test is not enforceable.
- The values of *P-value* in Tables C.1 and C.2 imply that PSOSCALF' performance is better compared with other PSO variants algorithms and other algorithms for unimodal functions F1–F7 and multimodal functions F8–F13 (except F9). In other words, these test values indicate that there is a significance difference between the PSOSCALF' performance and other algorithms.
- From the obtained results in Tables C.1 and C.2, it can be seen that for some fixed-dimension multimodal functions, performance of the proposed method is not significant better compared to other PSO family algorithms such as PSO, HPSO-TVAC, TACPSO, MPSO, AGPSO, HEPSO and LFPSO. The same results can be seen for GA, DA and MFO algorithms.
- For all the benchmark functions F1–F23, the results of the PSOSCALF algorithm are significantly better than SCA, ALO, FA and MVO algorithms in all the runs.

From the above, it can be concluded that the results of proposed method tend to be significantly better than the considered algorithms in here for most benchmark functions.

7. PSOSCALF for classical engineering problems

For evaluating a meta-heuristic method, a common approach is to implement the content of the method on to the real engineering problems. Unlike benchmark functions, the optimal value of most of the classic engineering problems is unknown. In addition, the real problems have many inequality and equality constraints. Therefore, evaluating the proposed method is considered necessary by constrained problems. In this section, PSOSCALF is tested with 8 constrained engineering problems: a gear train design, a pressure vessel, tension/compression spring, an I-beam problem,

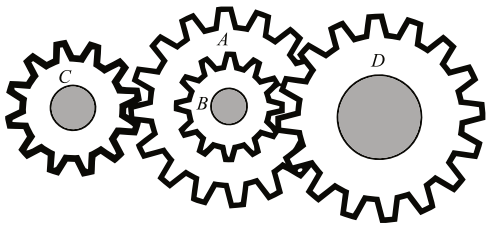

Fig. 8. Gear train design problem.

Table 5
Comparison result of the gear train design problem.

Algorithm	Optimal values for variables				Optimal gear ratio
	n_A	n_B	n_c	n_D	
PSOSCALF	49	19	16	43	2.7009e−12
CS [7]	43	16	19	49	2.7009e−12
ALO [41]	49	19	16	43	2.7009e−12
MFO [43]	43	19	16	49	2.7009e−12
MVO [44]	43	16	19	49	2.7009e−12
ISA [46]	N/A	N/A	N/A	N/A	2.7009e−12
MBA [48]	43	16	19	49	2.7009e−12
GA [49]	N/A	N/A	N/A	N/A	2.33e−7
GA [50]	33	14	17	50	1.362e−9
ABC [51]	19	16	44	49	2.78e−11
ALM [52]	33	15	13	41	2.1469e−8

a speed reducer, a three-bar truss and a welded beam, and inverse shape design in fluid flow.

Here, for solving these problems, we can use the various penalty functions described in [45]. Since the search for a type of penalty function is not the main subject of this paper, the simplest of them, the death penalty, is used. The implementation of this approach is very simple to solve the constrained optimization problems, and its computation volume is very low compared to other penalty functions [45].

7.1. Gear train design problem

One of the famous unconstrained optimization problem in mechanical engineering is gear train design problem. The schematic of this problem appears in Fig. 8. The main aims in this problem is to minimize the gear ratio of the set shown in Fig. 8.

The design variables of this optimization problem are tooth number of gears A, B, C, and D. [46,47]. The mathematical formula for this problem is as follows:

$$\begin{aligned} &\text{Consider} && \vec{x} = [x_1, x_2, x_3, x_4] = [n_A, n_B, n_C, n_D] \\ &\text{Minimize} && \mathrm{f}(\vec{x}) = \left(\frac{1}{6.931} - \frac{x_3 x_2}{x_1 x_4}\right)^2 \\ &\text{Subject to} && 12 \le x_1, x_2, x_3 \text{ and } x_4 \le 60 \end{aligned} \tag{19}$$

The gear train is a discrete problem. Before calculating the objective function, the variables obtained are round to the nearest integer. The best results from the proposed PSOSCALF method and the other various methods are provided in Table 5. We compared the results to CS [7], ALO [41], MFO [43], MVO [44], ISA [46], MBA [48], GA [49,50], ABC [51], and ALM [52]. As shown in this table, the optimal gear ratio resulting from the PSOSCALF algorithm is similar to the algorithms CS [7], ALO [41], MFO [43], ISA [46] and MBA [48] and the number of teeth obtained is similar to the ALO [41] algorithm. These results demonstrate the efficiency of the PSOSCALF method to solve the real discrete problems.

7.2. Pressure vessel design

The main objective of this problem is to minimize the construction costs of pressure vessel. The total construction cost includes welding, shaping and materials. On the other hand, these costs are influenced by the geometric parameters of this pressure vessel. Therefore, for having the least cost, it is necessary to determine the optimal values of the geometric parameters shown in Fig. 9. The mathematical model of this problem is formulated as follows [6]:

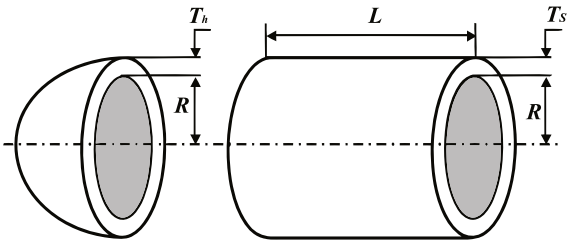

Fig. 9. Pressure design problem.

$$\begin{aligned} &\text{Consider} && \vec{x} = [x_1, x_2, x_3, x_4] = [T_s, T_h, R, L] \\ &\text{Minimize} && \mathrm{f}(\vec{x}) = 0.6224 x_1 x_3 x_4 + 1.7781 x_2 x_3^2 \\ & && +3.1661 x_1^2 x_4 + 19.84 x_1^2 x_3 \\ &\text{Subject to} && g_1(\vec{x}) = -x_1 + 0.0193 x_3 \le 0 \\ & && g_2(\vec{x}) = -x_3 + 0.00954 x_3 \le 0 \\ & && g_3(\vec{x}) = -\pi x_3^2 x_4 - \tfrac{4}{3}\pi x_3^3 + 1296000 \le 0 \\ & && g_4(\vec{x}) = x_4 - 240 \le 0 \\ &\text{Variable range} && 0 \le x_1 \text{ and } x_2 \le 99 \\ & && 10 \le x_3 \text{ and } x_4 \le 200 \end{aligned} \tag{20}$$

The results of solving this problem with the proposed PSOSCALF algorithm explained in this work and other algorithms such as GSA [6], WOA [6], MFO [43], MVO [44], Branch-Bound [47], Lagrangian multiplier [52], GWO [53], improved HS [54], PSO [55], GA [56–58], ES [59], DE [60], and ACO [61] presented in Table 6. The results of this table show that among all methods, the PSOSCALF method is able to provide the best design with the lowest cost.

7.3. Tension/compression spring design

Another problem used to evaluate different algorithms is design of the tension/compression spring. The main goal of this engineering problem is to find the optimal design parameters for minimizing the construction cost of a spring subjected to four nonlinear constraints. The design parameters in this problem are shown in Fig. 10. These parameters are: wire diameter (d), mean diameter of spring (D), number of active coils (N).

The objective function and constraints governing this problem are formulated as follows [6]:

$$\begin{aligned} &\text{Consider} && \vec{x} = [x_1, x_2, x_3] = [d, D, N] \\ &\text{Minimize} && \mathrm{f}(\vec{x}) = (x_3 + 2) x_2 x_1^2 \\ &\text{Subject to} && g_1(\vec{x}) = 1 - \frac{x_2^3 x_3}{71785 x_1^4} \le 0 \\ & && g_2(\vec{x}) = \frac{4x_2^2 - x_1 x_2}{125666(x_2 x_1^3 - x_1^4)} + \frac{1}{5108 x_1^2} \le 0 \\ & && g_3(\vec{x}) = 1 - \frac{140.45 x_1}{x_2^2 x_3} \le 0 \\ & && g_4(\vec{x}) = \frac{x_1 + x_2}{1.5} \le 0 \\ &\text{Variable range} && 0.05 \le x_1 \le 2 \\ & && 0.25 \le x_2 \le 1.3 \\ & && 2 \le x_3 \le 15 \end{aligned} \tag{21}$$

This nonlinear problem was optimized using optimization methods such as GSA [6], WOA [6], MFO [43], GWO [53], improved HS [54], PSO [55], GA [56], ES [59], DE [60], RO [62], mathematical optimization [63], and constraint correction [63]. The best results

Table 6
Comparison result for pressure vessel design problem.

Algorithm	Optimal values for variables				Optimal cost
	T_s	T_h	R	L	
PSOSCALF	1.25	0.0625	64.7668	11.9886	3137.3
GSA [6]	1.125	0.625	58.9886598	844542025	85388359
WOA [6]	0.8125	0.4375	42.0982699	176.638998	6059.7410
MFO [43]	0.8125	0.4375	42.098445	176.636596	6059.7143
MVO [44]	0.8125	0.4375	42.0907382	176.738690	6060.8066
Branch-Bound [45]	1.125	0.625	47.7	117.7	8129.1036
GWO [53]	0.8125	0.4345	42.089181	176.75731	6051.5639
Improved HS [54]	1.125	0.625	58.29015	43.69268	7197.730
PSO [55]	0.8125	0.4375	42.091266	176.7465	6061.0777
GA [56]	0.8125	0.4345	40.3239	200	6288.7445
GA [57]	0.8125	0.4375	42.097398	176.654050	6059.9463
GA [58]	0.9375	0.5	48.329	112.679	6410.3811
ES [59]	0.8125	0.4375	42.098087	176.640518	6059.7456
DE [60]	0.8125	0.4375	42.098411	176.637690	6059.7340
ACO [61]	0.8125	0.4375	42.103624	176.572656	6059.0888 (infeasible)
Lagrangian multiplier [52]	1.125	0.625	58.291	43.69	7198.0428

consistent with each of these methods are reported in Table 7. This table indicates that the PSOSCALF is more effective than other methods and provides the best design. Of course, PSOSCALF's results are very similar to the MFO [43] and GWO [53] methods.

7.4. Minimize the vertical deflection of an I-beam

Another problem that is considered here to evaluate the capabilities of the proposed method is the I-beam design problem (Fig. 11). The cost function of this structure is the vertical deflection of the beam and, as a result, to find the optimal geometric parameters related to the cross section of I-beam shaped. The design parameters in this problem are: length (b), height (h) and thicknesses t_f and t_w. These parameters are such that the cross section of the beam is smaller than or equal to 300 cm^2. Mathematical formula and constraints of this problem are as follows [43]:

$$\begin{aligned}
&\text{Consider} && \vec{x} = [x_1, x_2, x_3, x_4] = [b, h, t_w, t_f] \\
&\text{Minimize} && \mathrm{f}(\vec{x}) = \frac{5000}{\frac{t_w(h-2t_f)^3}{12} + \frac{bt_f^3}{6} + 2bt_f\left(\frac{h-t_f}{2}\right)^2} \\
&\text{Subject to} && g(\vec{x}) = 2bt_w + t_w(h-2t_f) \le 0 \\
&\text{Variable range} && 10 \le x_1 \le 50 \\
& && 10 \le x_2 \le 80 \\
& && 0.9 \le x_3 \text{ and } x_4 \le 5
\end{aligned} \tag{22}$$

The results of solving this nonlinear constraint problem using the PSOSCALF proposed method and other meta-heuristic algorithms such as CS [7], MFO [43], ARSM and IARSM [65], and SOS [66] are reported in Table 8. As shown in this table, the results of PSOSCALF and MFO [43] are similar and have obtained better solutions than other methods for this problem.

7.5. Speed reducer design

Another problem that has been solved by the PSOSCALF method and other optimization methods is the speed reducer. The schematic of this system is shown in Fig. 12.

The objective function in this problem is the construction cost of the speed reducer. The result of optimizing this function is to find the design parameters that are: face weight (b), module of teeth (m), number of teeth on pinion (z), the length of shaft 1 between bearings (l_1), the length of shaft 2 between bearings (l_2). This problem has a nonlinear objective function and nine non-linear inequalities and equalities that are formulated as

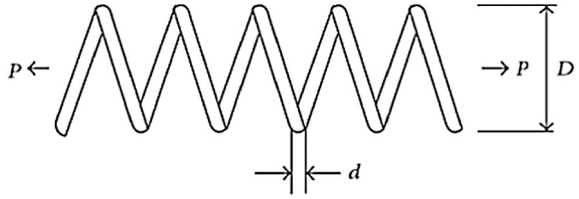

Fig. 10. Schematic of the tension/compression spring.

follows [7]:

$$\begin{aligned}
&\text{Consider} && \vec{x} = [x_1, x_2, x_3, x_4, x_5, x_6, x_7] \\
& && = [b, m, z, l_1, l_2, d_1, d_2] \\
&\text{Minimize} && \mathrm{f}(\vec{x}) = 0.7854x_1x_2^2\left(3.3333x_3^2\right. \\
& && \left.+14.9334x_3 - 43.0934\right) - 1.508b\left(x_6^2 + x_7^2\right) \\
& && +7.477\left(x_6^3 + x_7^3\right) + 0.7854(x_4x_6^2 + x_5x_7^2) \\
&\text{Subject to} && g_1(\vec{x}) = \frac{27}{x_1x_2^2x_3}P - 1 \le 0 \\
& && g_2(\vec{x}) = \frac{397.5}{x_1x_2^2x_3^2} - 1 \le 0 \\
& && g_3(\vec{x}) = \frac{1.93}{x_2x_3x_4^3x_6^4} - 1 \le 0 \\
& && g_4(\vec{x}) = \frac{1.93}{x_2x_3x_6^3x_7^4} - 1 \le 0 \\
& && g_5(\vec{x}) = \frac{\sqrt{\left(\frac{745x_4}{x_2x_3}\right)^2 + 1.69\times10^6}}{110x_6^3} - 1 \le 0 \\
& && g_6(\vec{x}) = \frac{\sqrt{\left(\frac{745x_6}{x_2x_3}\right)^2 + 157.5\times10^6}}{85x_7^3} - 1 \le 0 \\
& && g_7(\vec{x}) = \frac{x_2x_3}{40} - 1 \le 0 \\
& && g_8(\vec{x}) = \frac{5x_2}{x_1} - 1 \le 0 \\
& && g_9(\vec{x}) = \frac{x_1}{12x_2} - 1 \le 0 \\
&\text{Variable range} && 2.6 \le x_1 \le 3.6 \\
& && 0.7 \le x_2 \le 0.8 \\
& && 17 \le x_3 \le 28 \\
& && 7.3 \le x_4 \le 8.3 \\
& && 7.8 \le x_5 \le 8.3 \\
& && 2.9 \le x_6 \le 3.9 \\
& && 5 \le x_7 \le 5.5
\end{aligned} \tag{23}$$

The results of this optimization problem by using the PSOSCALF technique and different optimization methods are reported in

Table 7
Comparison of results for tension/compression spring design problem.

Algorithm	Optimum variables			Optimum weight
	D	*D*	*N*	
PSOCSCALF	0.0518836	0.36141614	11.018738	0.012665923
GSA [6]	0.050276	0.323680	13.525410	0.0127022
WOA [6]	0.051207	0.345215	12.004032	0.0126763
MFO [43]	0.051004457	0.36410932	10.868421862	0.0126669
GWO [53]	0.05169	0.356737	11.28885	0.012666
Improved HS [54]	0.051154	0.349871	12.076432	0.0126706
PSO [55]	0.051728	0.357644	11.244543	0.0126747
GA [56]	0.051480	0.351661	11.632201	0.0127048
ES [59]	0.051989	0.363965	10.890522	0.0126810
DE [60]	0.051609	0.354714	11.410831	0.0126702
RO [62]	0.051370	0.349096	11.76279	0.0126788
Mathematical optimization [63]	0.053396	0.399180	9.1854000	0.0127303
Constraint correction [64]	0.05	0.3159	14.25	0.0128334

Table 8
Comparison results for I-beam design problem.

Algorithm	Optimal values for variables				Optimal cost
	b	*h*	t_w	t_f	
PSOSCALF	50	80	1.7647	5	0.0066259
CS [7]	50	80	0.9	2.321675	0.0130747
MFO [43]	50	80	1.7647	5	0.0066259
ARSM [65]	37.05	80	1.71	2.31	0.0157
IARSM [65]	48.42	79.99	0.9	2.4	0.131
SOS [66]	50	80	0.9	2.32179	0.0130741

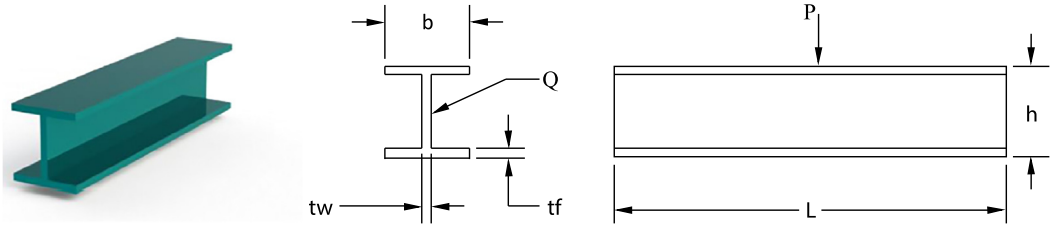

Fig. 11. I-beam design problem.

Table 9. The superiority of the obtained design with PSOSCALF technique in comparison with other optimization methods is seen clearly in this table. On the other hand, the solutions obtained by TAS [67] and SBS [68] methods violated the constraints and are infeasible. Therefore, CS method is ranked second.

7.6. Truss design problem with three-bar

One of the famous problems in the design of engineering structures, especially in civil engineering, is the three-bar truss design problem. The objective function is to minimize the truss weight so that it does not violate the tension, deformation and buckling constraints. In other words, this fitness function is equivalent to determine the optimal values for cross-sections of A_1 and A_2, taking into account the constraints. Parameters related to this problem are shown in Fig. 13.

The mathematical formula for this problem is given below [44]:

$$\begin{aligned} &\text{Consider} && \vec{x} = [x_1, x_2] = [A_1, A_2] \\ &\text{Minimize} && f(\vec{x}) = \left(2\sqrt{2}x_1 + x_2\right) l \\ &\text{Subject to} && g_1(\vec{x}) = \frac{\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1x_2} P - \sigma \le 0 \\ &&& g_2(\vec{x}) = \frac{x_2}{\sqrt{2}x_1^2 + 2x_1x_2} P - \sigma \le 0 \\ &&& g_3(\vec{x}) = \frac{1}{\sqrt{2}x_2 + x_1} P - \sigma \le 0 \end{aligned} \quad (24)$$

$$\text{Variable range} \quad 0.05 \le x_1, x_2 \le 2 \quad (25)$$

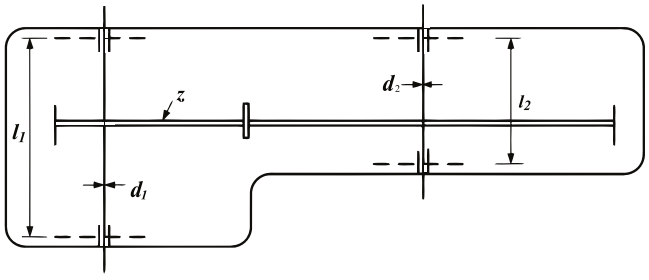

Fig. 12. Speed reducer.

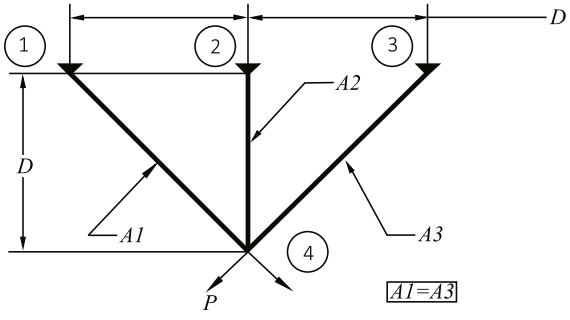

Fig. 13. Three-bar truss design problem.

where $l = 100$ cm, $P = 2$ KN/cm^2, $\sigma = 2$ KN/cm^2

Table 9
Comparison results for speed reducer design.

Algorithm	Optimal values for variables							Optimal cost
	b	M	Z	l_1	l_2	d_1	d_2	
PSOCSCALF	3.5	0.7	17	7.3	7.8	2.9	5.2865	1909.9355
CS [7]	3.5015	0.7	17	7.6050	7.8181	3.3520	5.2875	3000.9810
TAS [67]	3.6	0.7	17	7.3	7.8	3.4	5	2876.117623 (infeasible)
SBS [68]	3.506122	0.700006	17	7.549126	7.85933	3.365576	5.289773	3008.08
Ray and Saini [69]	3.514185	0.700005	17	7.497343	7.8346	2.9018	5.0022	2732.9006 (infeasible)
Montes and Coello [70]	3.506163	0.700831	17	7.460181	7.962143	3.3629	5.3090	3025.005

Table 10
Comparison results of the three-bar truss design problem.

Algorithm	Optimal values for variables		Optimal Cost
	A_1	A_2	
PSOCSCALF	0.788662245623587	0.408284747203991	263.89584349
CS [7]	0.78867	0.40902	263.9716
ALO [41]	0.788662816000317	0.408283133832901	263.8958434
MFO [43]	0.788244770931922	0.409466905784741	263.985979682
MVO [44]	0.78860276	0.40845307	263.8958499
MBA [48]	0.788565	0.4085597	263.8958522
DEDS [71]	0.78867513	0.40824828	263.8958434
Tsa [72]	0.788	0.408	263.68
PSO-DE [73]	0.7886751	0.4082482	263.8958433

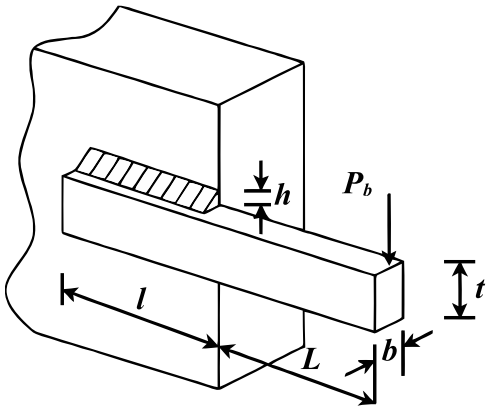

Fig. 14. Schematic of welded beam design.

In spite of the fact that this problem is very simple in implementing, it has been considered by many researchers and has been solved by various meta-heuristic algorithms. For example, CS [7], ALO [41], MFO [43], MVO [44], MBA [48], DEDS [71], Tsa [72], and PSO-DE [73]. The results of the PSOSCALF algorithm with these algorithms are compared in Table 10. As can be seen, PSOSCALF results are very close to the results of the ALO [41], MVO [44], DEDS [71], and PSO-DE [73] algorithms. This indicates that the PSOSCALF algorithm can effectively resolve non-linear constrained problems. It can be observed from Table 10 that the results of the Tsa method violate one of the constraints, so the obtained design by this method is infeasible [72].

7.7. Welded beam design problem

Similar to above examples, the fitness function of this problem is the construction cost of the welded beam as shown in Fig. 14. The constraints governing this problem include: shear stress (τ), bending stress (σ), buckling load on the beam (P_c) and end deflection of the beam (δ). This problem has four parameters: weld thickness (h), the clamped bar length (L), the bar height (t) and the bar thickness (b).

This nonlinear optimization problem is formulated as follows [6]:

$$
\begin{aligned}
&\text{Consider} && \vec{x} = [x_1, x_2, x_3, x_4] = [h, L, T, b] \\
&\text{Minimize} && f(\vec{x}) = 1.10471 x_1^2 x_2 + 0.04811 x_3 x_4 (14 + x_2) \\
&\text{Subject to} && g_1(\vec{x}) = \tau(\vec{x}) - \tau_{\max} \le 0 \\
& && g_2(\vec{x}) = \sigma(\vec{x}) - \sigma_{\max} \le 0 \\
& && g_3(\vec{x}) = \delta(\vec{x}) - \delta_{\max} \le 0 \\
& && g_4(\vec{x}) = x_1 - x_4 \le 0 \\
& && g_5(\vec{x}) = P - P_c(\vec{x}) \le 0 \\
& && g_6(\vec{x}) = 0.125 - x_1 \le 0 \\
& && g_7(\vec{x}) = 1.10471 x_1^2 + 0.04811 x_3 x_4 (14 + x_2) \\
& && -5 \le 0 \\
&\text{Variable range} && 0.1 \le x_1, x_4 \le 2 \\
& && 0.1 \le x_2, x_3 \le 10
\end{aligned}
\tag{26}
$$

where $P = 6000\,lb, L = 14\,in, \delta_{\max} = 0.25in, \tau_{\max} = 13\,600psi, \sigma_{\max} = 30\,000psi$. Also, full details for $\tau(\vec{x})$, $\sigma(\vec{x})$ and $\delta(\vec{x})$ parameters can be found in [6].

This problem has been solved with various optimization methods in accordance with Table 11 and compared with the results of the PSOSCALF method. The results of Table 11 show that the obtained design by the proposed PSOSCALF algorithm does not violate constraints and has the lowest fabrication cost compared to other numerical methods, and MFO [43] and CBO [74] methods are ranked second and third, respectively.

7.8. Inverse shape design in fluid flow problems

As the last classical test case, in this subsection, the PSOSCALF method is applied for inverse shape design of a typical cylinder and a symmetric airfoil in external flow problems. The shape design problem process is performed in the context of incompressible, in viscid (ideal) flow in terms of secondary variables, i.e. stream function. In this state, the governing equation is expressed as:

$$\nabla^2 \psi = 0 \tag{27}$$

Table 11
Comparison of Results for the Welded Beam Design Problem.

Algorithm	Optimal values for variables				Optimal cost
	H	*L*	*T*	*b*	
PSOSCALF	0.140946034	2.8644127185	9.036471193	0.20573659	1.57126326
WOA [6]	0.205396	3.484293	9.037426	0.206276	1.730499
GSA [6]	0.182129	3.8569979	10.00000	0.202376	1.8799522
MFO [43]	0.2057	3.4703	9.0364	0.2057	1.72452
MVO [44]	0.205463	3.473193	9.044502	0.205695	1.72645
GWO [53]	0.205676	3.478377	9.03681	0.205778	1.72624
Improved HS [54]	0.20573	3.47049	9.03662	0.2057	1.7248
RO [62]	0.203687	3.528467	9.004233	0.207241	1.735344
CBO [74]	0.205722	3.47041	9.037276	0.205735	1.724663
GA [75]	N/A	N/A	N/A	N/A	1.8245
GA [76]	N/A	N/A	N/A	N/A	2.3800
GA [77]	0.2489	6.1730	8.1789	0.2533	2.4331
HS [78]	0.2442	6.2231	8.2915	0.2443	2.3807
Random [79]	0.4575	4.7313	5.0853	0.6600	4.1185
Simplex [79]	0.2792	5.6256	7.7512	0.2796	2.5307
David [79]	0.2434	6.2552	8.2915	0.2444	2.3841
APPROX [79]	0.2444	6.2189	8.2915	0.2444	2.3815
CPSO [80]	0.202369	3.544214	9.048210	0.205723	1.73148

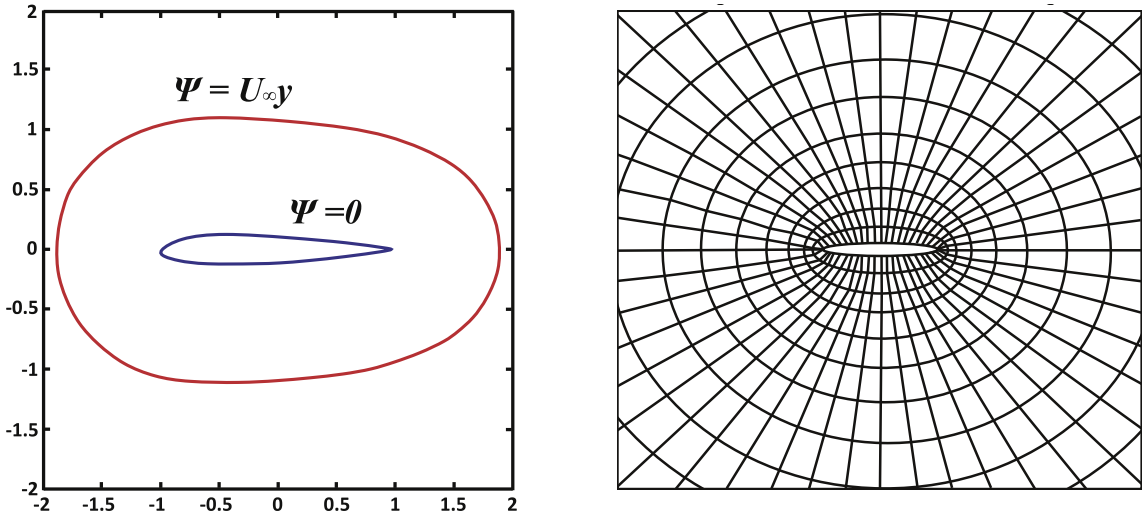

Fig. 15. Inverse shape design of bodies in external incompressible possible flow problems (a) auxiliary boundary conditions (b) employed quadrilateral structured grids.

A general schematic of the problem and the auxiliary boundary conditions are shown in Fig. 15a. Eq. (28) requires to be discretized and solved numerically on the physical domain. Here, an element-based finite volume method approach [81–85] is employed to discretize the governing equation in which after mesh generation, a control volume is associated with each node. A schematic of employed computational quadrilateral structured grids is depicted in Fig. 15b.

Numerical solution of the governing equation reveals the stream function distribution in the physical domain. Thereafter, the tangential velocity along the surface of the body is obtained as follows:

$$V_{tan} = \frac{\partial \psi}{\partial N} \tag{28}$$

Here, the method is applied by a so-called reproduction procedure in which the target geometry is already known. In other words, as the first problem, the governing equation in a physical domain containing cylinder is solved. The tangential velocity around the surface of the cylinder is saved as target function. In the next step, the target function is introduced to the PSOSCALF method to find to corresponding geometry. Results are provided in Fig. 16a. As it can be seen, the employed PSOSCALF method has captured the target geometry i.e. a cylinder with radius equal to 1.5 (r = 1.5) in excellent agreement. Note that, due to symmetrical geometry, the shape reproduction procedure is carried out for half of the body by 31 radius as 31 design variables which could vary between $0.5 \leq r_i \leq 2$, i = 1,2,..,31.

As the second problem, the PSOSCALF method is applied for NACA0012 airfoil inverse shape design. Similar to the previous test case, tangential velocity distribution along the surface of the NACA0012 airfoil is introduced to the PSOSCALF method. In this case, instead of radius, the thickness of the different parts of the airfoil is considered as design variables. Results of the inverse design is reported in Fig. 16b. NACA0012 airfoil with chord length equal to 5 (C = 5) and thickness equal to 0.5 (th = 0.5) is considered as target geometry. It is evident that the PSOSCALF method performance in capturing the target geometry in this case is greatly acceptable.

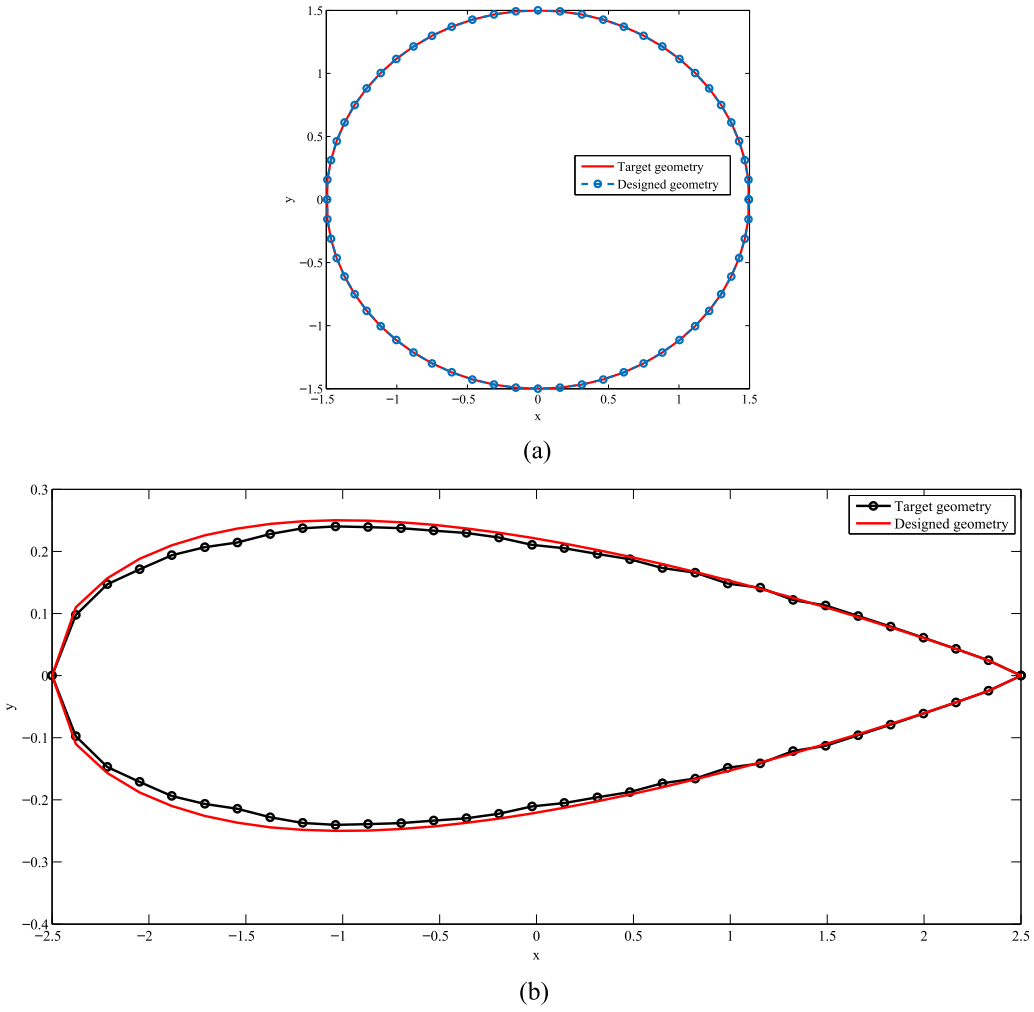

Fig. 16. Designed geometries in external incompressible potential flows via PSOSCALF method (a) inverse shape design of cylinder (b) inverse shape design of airfoil.

8. Conclusion

In this study, in order to solve the famous weakness points of PSO algorithm of early convergence and being trapped in the local minimum, the PSO algorithm was combined with the position updating equations of the SCA algorithm and the Levy flight, and a new approach called PSOSCALF was created. Then, the PSOSCALF performance was compared with the other PSO variants and the famous algorithms proposed in recent years. For this purpose, 23 benchmark functions and 8 nonlinear constrained problems were used. In this study, unimodal and multimodal test functions were used. The evaluation of the results of the optimization of the benchmark functions showed that the PSOSCALF method has an appropriate convergence rate and is more successful in most of the functions in finding the global optima than other algorithms. In fact, by utilizing the benefits of the SCA method and Levy flight, two features are created in the new approach in this article:

(1) The formation of equilibrium between the two phases of exploration and exploitation, which leads to improve the convergence behavior of the PSO optimization algorithm.
(2) Increasing the global search capability and the probability of finding the global optima.

On the other hand, the results of the design engineering problems showed that the PSOSCALF algorithm has a high performance in the unknown search space. In fact, these results prove the capability of the PSOSCALF approach in optimizing the real constrained problems and discrete problems with different characteristics.

Acknowledgments

We express our sincere gratitude to P. Mayeli (Monash University) for his helpful and constructive suggestions in solving inverse shape design in fluid flow problems. We also thank Z. Ranjbar (Ahrar Institute of Technology and Higher Education) for her assistance in preparation of the manuscript. The authors appreciate the anonymous reviewers and the editor for their valuable comments and suggestions for improving this paper.

Appendix A. Results of optimizing the benchmark functions by PSOSCALF and Other PSO variants

See Tables A.1–A.3.

Appendix B. Results of optimizing the benchmark functions by PSOSCALF and other algorithms

See Tables B.1–B.3.

Appendix C. Results of Wilcoxon's Rank Sum Test

See Tables C.1 and C.2.

Table A.1
Comparison of PSOSCALF with PSO variants for the unimodal functions.

F		PSO	HPSO-TAVC	TACPSO	MPSO	AGPSO	HEPSO	RPSOLF	LFPSO	PSOSCALF
F1	Mean	1.4407e−08	5.18808e−07	9.11126e−04	8.14859e−07	1.16089e+04	16.26772	**5.06503e−269**	1.06278e−04	1.11014e−20
	Std.dev.	3.0815e−08	2.18659e−06	0.00247	1.03565e−06	2.48543e+03	10.01293	**0**	1.58804e−04	1.83289e−20
	Rank	3	4	7	5	9	8	**1**	6	2
F2	Mean	0.00017	0.01907	0.00757	5.97782e−04	0.00261	1.28424	**1.00002e−134**	0.00173	4.09460e−11
	Std.dev.	0.00049	0.06617	0.01546	0.00117	0.00481	0.41611	**3.75391e−134**	0.00453	5.68981e−11
	Rank	3	8	7	4	6	9	**1**	5	2
F3	Mean	71.66699	2.35214e+02	41.84664	41.90292	13.84101	7.42375e+03	**7.79182e−249**	1.18273e+03	2.16858e−12
	Std.dev.	45.59262	1.76838e+02	33.37176	31.37381	14.54349	3.87991e+03	**0**	5.66036e+02	1.03815e−11
	Rank	6	7	4	5	3	9	**1**	8	2
F4	Mean	1.188414	5.09682	1.32096	1.03002	1.82377	23.95145	**1.93773e−157**	12.15124	8.47410e−08
	Std.dev.	0.54926	1.39444	0.42809	0.48370	0.83725	7.71460	**1.06134e−156**	6.22665	1.23324e−07
	Rank	4	7	5	3	6	9	**1**	8	2
F5	Mean	35.10321	74.05163	44.31173	51.65164	33.09881	2.380365e+03	27.42672	97.82509	**21.97646**
	Std.dev.	22.18090	48.59594	31.18012	35.69627	27.21112	1.852345e+03	0.24848	65.33289	**0.54774**
	Rank	4	7	5	6	3	9	2	8	**1**
F6	Mean	1.3861e−08	1.69374e−07	4.55628e−04	2.29072e−06	2.96339e−10	21.55405	2.98244	1.02748e−04	**7.13998e−12**
	Std.dev.	1.9479e−08	3.97164e−07	5.85416e−04	3.35300e−06	1.25187e−09	9.33263	0.23250	1.21085e−04	**3.65884e−11**
	Rank	3	4	7	5	2	9	8	6	**1**
F7	Mean	0.01378	0.11504	0.01560	0.01049	0.02147	0.12982	0.00104	0.04354	**0.00012**
	Std.dev.	0.00413	0.05692	0.00701	0.00441	0.01002	0.09727	7.64408e−04	0.01113	**0.00010**
	Rank	4	8	5	3	6	9	2	7	**1**
	Final Rank	3	7	6	4	5	9	2	8	**1**

Table A.2
Comparison of PSOSCALF with PSO variants for the multimodal functions.

F		PSO	HPSO-TAVC	TACPSO	MPSO	AGPSO	HEPSO	RPSOLF	LFPSO	PSOSCALF
F8	Mean	−6451.4871	−9.82506e+03	−7.19718e+03	−6.87074e+03	−6.87883e+03	−2.13912e+11	−3.25452e+03	−8.74923e+03	**−12569.48661**
	Std.dev.	768.9100	1.72188e+03	6.37766e+02	5.66229e+02	7.73983e+02	8.282171e+11	2.86089e+02	5.46109e+02	**2.39996e−07**
	Rank	7	2	4	6	5	9	8	3	**1**
F9	Mean	43.71181	20.69976	35.12248	37.04561	35.28784	42.00118	**0**	29.69942	**0**
	Std.dev.	10.26013	5.38991	8.68230	8.95531	12.20452	7.08632	**0**	4.29280	**0**
	Rank	8	2	4	6	5	7	**1**	3	**1**
F10	Mean	0.37818	4.82870	0.17989	0.43132	1.47012	2.83842	**4.08562e−15**	0.02995	2.24609e−11
	Std.dev.	0.62463	1.08039	0.46217	0.70951	0.71970	0.66134	**1.08403e−15**	0.11839	2.335472e−11
	Rank	5	9	4	6	7	8	**1**	3	2
F11	Mean	0.01663	0.23057	0.01535	0.00959	0.02263	1.16858	**0**	0.01138	**0**
	Std.dev.	0.01997	0.14039	0.01517	0.01095	0.02733	0.12602	**0**	0.01617	**0**
	Rank	6	8	5	3	7	9	**1**	4	**1**
F12	Mean	0.06910	0.19379	0.02419	0.02073	0.01728	0.47856	0.26157	0.29179	**8.46465e−14**
	Std.dev.	0.11333	0.36667	0.05224	0.06325	0.03929	0.22623	0.03386	0.65905	**2.79106e−13**
	Rank	5	6	4	3	2	9	7	8	**1**
F13	Mean	0.04652	0.27040	0.00531	0.00732	0.00656	1.85056	2.05282	0.01337	**0.00399**
	Std.dev.	0.22490	0.76354	0.00558	0.02126	0.01835	0.65246	0.16579	0.02597	**0.00928**
	Rank	6	7	2	4	3	8	9	5	**1**
	Final Rank	8	7	2	5	6	9	4	3	**1**

Table A.3
Comparison of PSOSCALF with PSO variants for the fixed-dimension multimodal functions.

F		PSO	HPSO-TAVC	TACPSO	MPSO	AGPSO	HEPSO	RPSOLF	LFPSO	PSOSCALF
F14	Mean	3.29771	1.03113	1.13054	1.16236	1.13054	**0.99800**	1.54064	**0.99800**	1.13027
	Std.dev.	2.62747	0.18148	0.34368	0.90024	0.34368	**9.21990e−17**	1.84429	**9.21990e−17**	0.50338
	Rank	7	2	4	6	4	**1**	5	**1**	3
F15	Mean	0.00184	4.15499e−04	3.41464e−04	3.84379e−04	0.00107	6.40451e−04	0.00171	0.00118	**3.13244e−04**
	Std.dev.	0.00505	3.03811e−04	1.86107e−04	2.94709e−04	0.00365	2.80158e−04	0.00508	0.00363	**2.17489e−05**
	Rank	9	4	2	3	6	5	8	7	**1**
F16	Mean	−1.03162	−1.03162	−1.03162	−1.03162	−1.03162	−1.03162	−1.03161	−1.03162	**−1.0316**
	Std.dev.	6.77521e−16	6.11579e−16	6.51945e−16	6.38773e−16	6.38773e−16	3.55486e−15	1.65038e−05	6.51945e−16	**4.40244e−16**
	Rank	6	3	5	4	4	7	2	5	**1**
F17	Mean	**0.39788**	0.39788	**0.39788**	**0.39788**	**0.39788**	0.39788	0.39837	**0.39788**	0.39788
	Std.dev.	**0**	3.24338e−16	**0**	**0**	**0**	6.59474e−13	5.26711e−04	**0**	3.66527e−15
	Rank	**1**	2	**1**	**1**	**1**	4	5	**1**	3
F18	Mean	2.99999	2.99999	2.99999	2.99999	2.99999	3	3.00002	2.99999	**3**
	Std.dev.	6.99741e−16	3.06455e−15	1.67994e−15	1.02668e−15	1.88771e−15	5.14644e−11	1.65837e−05	1.65342e−15	**5.96540e−13**
	Rank	3	8	6	4	7	2	9	5	**1**
F19	Mean	−3.86278	−3.86278	−3.86278	−3.86278	−3.86278	−3.86278	**−3.85923**	−3.86278	−3.86278
	Std.dev.	2.68234e−15	2.43935e−15	2.66836e−15	2.64017e−15	2.64017e−15	1.00875e−13	**0.00283**	2.66836e−15	8.31755e−15
	Rank	4	2	4	3	3	6	**1**	4	5
F20	Mean	−3.27443	−3.27047	−3.27840	−3.28236	−3.27840	**−3.31803**	−3.10441	−3.27840	−3.27168
	Std.dev.	0.05924	0.05992	0.05827	0.05700	0.05827	**0.02170**	0.15760	0.05827	0.06371
	Rank	4	6	3	2	3	**1**	7	3	5
F21	Mean	−5.73236	−7.82253	−7.64690	−8.72528	−8.06257	−10.15319	−4.76171	−8.28850	**−10.15319**
	Std.dev.	3.31956	3.40044	3.21814	2.44474	3.09122	2.68028e−05	0.73723	2.742524	**4.46227e−15**
	Rank	8	6	7	3	5	2	9	4	**1**
F22	Mean	−6.88146	−8.20240	−8.95383	−9.87415	−9.39650	−10.39978	−4.81927	−9.97210	**−10.40294**
	Std.dev.	3.65169	3.23277	2.97492	1.61348	2.32408	0.01728	0.75699	1.66904	**1.80672e−15**
	Rank	8	7	6	4	5	2	9	3	**1**
F23	Mean	−6.84098	−9.35303	−10.08722	−10.3577	−9.54436	−10.53640	−5.06376	−10.10220	**−10.53640**
	Std.dev.	3.79229	2.72218	1.74724	0.97873	2.57736	5.84580e−07	0.82968	1.67988	**4.84794e−15**
	Rank	8	7	5	3	6	2	9	4	**1**
	Final Rank	8	7	5	3	6	2	9	4	**1**

Table B.1
Comparison of PSOSCALF with other algorithms for the unimodal functions.

F		GA	WOA	DA	SCA	ALO	FA	MFO	MVO	PSOSCALF
F1	Mean	0.63300	**3.71976e−84**	1.88350e+03	3.5881	1.19658e−04	2.11699e+04	2.33483e+03	0.73132	1.11014e−20
	Std.dev.	0.25700	**1.75152e−83**	2.69839e+03	7.0558	5.46848e−05	3.55470e+03	4.30131e+03	0.16252	1.83289e−20
	Rank	4	**1**	7	6	3	9	8	5	2
F2	Mean	0.17875	**5.53162e−54**	13.05423	0.00461	38.32340	1.78590e+02	32.41720	1.32301	4.09460e−11
	Std.dev.	0.05905	**2.32454e−53**	6.41237	0.00538	47.22503	2.21084e+02	17.31901	3.97493	5.68981e−11
	Rank	4	**1**	6	3	8	9	7	5	2
F3	Mean	3.91141e+03	2.79218e+04	1.78574e+04	6.19658e+03	1.91786e+03	2.78278e+04	1.92956e+04	87.65831	**2.16858e−12**
	Std.dev.	1.62928e+03	1.20258e+04	1.07885e+04	4.28035e+03	9.38004e+02	7.75841e+03	1.28812e+04	40.76549	**1.03815e−11**
	Rank	4	9	6	5	3	8	7	2	**1**
F4	Mean	4.80456	39.30104	34.17868	26.86848	12.57920	52.96710	58.69915	1.24477	**8.47410e−08**
	Std.dev.	1.23984	25.33288	15.30780	9.70965	3.99733	4.98726	9.39231	0.49953	**1.23324e−07**
	Rank	3	7	6	5	4	8	9	2	**1**
F5	Mean	2.22248e+02	27.56931	5.44199e+05	2.14543e+04	2.04769e+02	2.72752e+04	2.67774e+06	2.83888e+02	**21.97646**
	Std.dev.	1.67410e+02	0.52686	5.61474e+05	5.01328e+04	4.25635e+02	2.08100e+04	1.45932e+07	4.48486e+02	**0.54774**
	Rank	4	2	8	6	3	7	9	5	**1**
F6	Mean	0.55122	0.06690	1.99419e+03	17.69731	9.98645e−05	2.14498e+04	1.33671e+03	0.68486	**7.13998e−12**
	Std.dev.	0.26780	0.05826	2.95027e+03	34.65970	5.98309e−05	3.63143e+03	3.45671e+03	0.18549	**3.65884e−11**
	Rank	4	3	8	6	2	9	7	5	**1**
F7	Mean	0.05477	0.00149	1.04726	0.04917	0.13015	0.27544	2.56529	0.02251	**0.00012**
	Std.dev.	0.01890	0.00233	2.87437	0.02761	0.04475	0.11814	4.80338	0.01063	**0.00010**
	Rank	5	2	8	4	6	7	9	3	**1**
	Final Rank	4	2	6	5	3	8	7	3	**1**

Table B.2
Comparison of PSOSCALF with other algorithms for the multimodal functions.

F		GA	WOA	DA	SCA	ALO	FA	MFO	MVO	PSOSCALF
F8	Mean	−1.09985e+04	−1.12873e+04	−5.92666e+03	−3.91973e+03	−6.02918e+03	−7.23113e+03	−8.52834e+03	−7.78451e+03	**−12569.48661**
	Std.dev.	2.85868e+02	1.56509e+03	9.04236e+02	2.84979e+02	1.526108e+03	7.21432e+02	8.44490e+02	6.53422e+02	**2.39996e−07**
	Rank	3	2	8	9	7	6	4	5	**1**
F9	Mean	0.35146	**0**	1.41046e+02	35.11823	73.75954	2.01677e+02	1.53555e+02	1.164211e+02	**0**
	Std.dev.	0.27963	**0**	47.87805	32.80977	18.63907	36.73670	34.01286	30.05497	**0**
	Rank	2	**1**	6	3	4	8	7	5	**1**
F10	Mean	0.19637	**4.20404e−15**	19.90270	13.09973	2.76566	18.91265	14.68451	1.36065	2.24609e−11
	Std.dev.	0.06895	**2.45667e−15**	0.09087	9.01308	1.51403	0.22637	7.35687	0.81735	2.335472e−11
	Rank	3	**1**	9	6	5	8	7	4	2
F11	Mean	0.47928	**0**	17.589217	0.69997	0.02627	3.62007	15.84481	0.73701	**0**
	Std.dev.	0.15714	**0**	21.99561	0.40542	0.01952	54.63988	34.03455	0.09224	**0**
	Rank	3	**1**	8	4	2	6	7	5	**1**
F12	Mean	0.00239	0.00731	3.34611e+05	1.51306e+04	10.59120	9.54951e+03	5.34522	1.44471	**8.46465e−14**
	Std.dev.	0.00340	0.00912	1.54348e+06	8.16749e+04	5.06278	1.95986e+04	4.32243	1.11730	**2.79106e−13**
	Rank	2	3	9	8	6	7	5	4	**1**
F13	Mean	0.03533	0.23382	6.02465e+05	2.32170e+05	6.36878	5.79349e+05	13.02448	0.12553	**0.00399**
	Std.dev.	0.01931	0.12955	1.33452e+06	1.21373e+06	9.89703	5.41371e+05	20.26428	0.06345	**0.00928**
	Rank	2	4	9	7	5	8	6	3	**1**
	Final Rank	3	2	9	7	5	8	6	4	**1**

Table B.3
Comparison of PSOSCALF with other algorithms for the fixed-dimension multimodal functions.

F		GA	WOA	DA	SCA	ALO	FA	MFO	MVO	PSOSCALF
F14	Mean	0.99800	2.33847	1.46083	1.72083	1.72459	8.76716	1.88908	**0.99800**	1.13027
	Std.dev.	1.40820e−10	2.94127	0.89141	1.88738	1.18772	6.09174	1.38094	**1.92268e−11**	0.50338
	Rank	2	8	4	5	6	9	7	**1**	3
F15	Mean	0.00162	6.75652e−04	0.00249	9.81174e−04	0.00420	0.00256	0.00107	0.00406	**3.13244e−04**
	Std.dev.	0.00136	4.54643e−04	0.00362	3.40514e−04	0.01221	0.00605	3.60168e−04	0.00741	**2.17489e−05**
	Rank	5	2	6	3	9	7	4	8	**1**
F16	Mean	−1.03162	−1.03162	−1.031628	−1.031597	−1.03162	−0.89560	−1.03162	−1.03162	**−1.0316**
	Std.dev.	1.44151e−06	7.18262e−11	3.28983e−07	3.16821e−05	6.31163e−14	0.30936	6.77521e−16	2.25617	**4.40244e−16**
	Rank	3	3	3	2	3	4	3	3	**1**
F17	Mean	0.39788	0.39788	0.39788	0.39887	0.39788	0.39788	**0.39788**	0.39788	0.39788
	Std.dev.	7.01954e−07	3.67701e−06	1.90643e−09	0.00104	1.28463e−14	1.58018e−12	**0**	7.20550e−08	3.66527e−15
	Rank	7	8	5	9	3	4	**1**	6	2
F18	Mean	3	3.00000	2.99999	3.00003	**3**	3	2.99999	3	3
	Std.dev.	4.93729e−05	2.12485e−05	4.33315e−15	4.39184e−05	**1.85937e−13**	4.49960e−11	1.56466e−15	1.50378e−06	5.96540e−13
	Rank	7	5	9	6	**1**	3	8	4	2
F19	Mean	−3.86278	**−3.85858**	−3.85098	−3.85545	−3.86278	−3.86278	−3.86278	−3.86278	−3.86278
	Std.dev.	1.44359e−07	**0.00685**	0.06352	0.00277	2.89417e−14	3.19194e−12	2.71008e−15	6.00496e−07	8.31755e−15
	Rank	6	**1**	9	8	4	5	2	7	3
F20	Mean	**−3.29028**	−3.27339	−3.17703	−3.00812	−3.28232	−3.28236	−3.21637	−3.27794	−3.27168
	Std.dev.	**0.05347**	0.07844	0.16086	0.11698	0.05705	0.05700	0.05194	0.05888	0.06371
	Rank	**1**	5	8	9	3	2	7	4	6
F21	Mean	−6.48612	−8.96954	−8.79364	−2.90092	−6.12008	−6.04625	−8.72901	−8.04689	**−10.15319**
	Std.dev.	3.57536	2.45750	2.29310	1.74510	2.84319	3.12032	2.69256	2.65812	**4.46227e−15**
	Rank	6	2	3	9	7	8	4	5	**1**
F22	Mean	−8.06141	−7.71398	−8.91052	−4.02160	−7.30542	−5.39930	−8.91931	−8.58003	**−10.40294**
	Std.dev.	3.39594	3.40709	2.54882	1.74729	3.24345	3.20939	2.77874	2.89763	**1.80672e−15**
	Rank	5	6	3	9	7	8	2	4	**1**
F23	Mean	−8.24148	−7.13216	−8.19558	−4.55529	−7.63839	−5.06227	10.00826	−8.23796	**−10.53640**
	Std.dev.	3.57812	3.50631	2.72152	1.21606	3.66626	3.68293	2.01056	3.15534	**4.84794e−15**
	Rank	3	7	5	9	6	8	2	4	**1**
	Final Rank	5	4	6	7	3	8	2	4	**1**

Table C.1
Comparison of PSOSCALF with PSO variants using Wilcoxon's rank sum test.

PSOSCALF Vs	Wilcoxon's rank sum test	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12
PSO	*P*-value	3.0198e−11	3.0198e−11	3.0198e−11	3.0198e−11	1.2493e−05	3.0198e−11	3.0198e−11	3.0066e−11	1.7202e−12	3.0198e−11	5.6744e−10	3.0198e−11
	h-value	1	1	1	1	1	1	1	1	1	1	1	1
	z-value	−6.6455	−6.6455	−6.6455	−6.6455	−4.3687	−6.6455	−6.6455	−6.6462	−7.0554	−6.6455	−6.1992	−6.6455
HPSO-TVAC	*P*-value	3.0198e−11	3.0198e−11	3.0198e−11	3.0198e−11	2.0022e−06	3.0198e−11	3.0198e−11	5.5513e−10	1.7202e−12	3.0198e−11	1.9603e−11	3.0198e−11
	h-value	1	1	1	1	1	1	1	1	1	1	1	1
	z-value	−6.6455	−6.6455	−6.6455	−6.6455	−4.7531	−6.6455	−6.6455	−6.2026	−7.0554	−6.6455	−6.7089	−6.6455
TACPSO	*P*-value	3.0198e−11	3.0198e−11	3.0198e−11	3.0198e−11	5.5726e−10	3.0198e−11	3.0198e−11	3.0066e−11	1.7202e−12	3.0198e−11	5.4160e−11	3.0198e−11
	h-value	1	1	1	1	1	1	1	1	1	1	1	1
	z-value	−6.6455	−6.6455	−6.6455	−6.6455	−6.2020	−6.6455	−6.6455	−6.6462	−7.0554	−6.6455	−6.5590	−6.6455
MPSO	*P*-value	3.0198e−11	3.0198e−11	3.0198e−11	3.0198e−11	2.1958e−07	3.0198e−11	3.0198e−11	3.0066e−11	1.7202e−12	3.0198e−11	1.7810e−10	3.0198e−11
	h-value	1	1	1	1	1	1	1	1	1	1	1	1
	z-value	−6.6455	−6.6455	−6.6455	−6.6455	−5.1819	−6.6455	−6.6455	−6.6462	−7.0554	−6.6455	−6.3791	−6.6455
AGPSO	*P*-value	3.0198e−11	3.0198e−11	3.0198e−11	3.0198e−11	0.0015	3.3383e−11	3.0198e−11	3.0066e−11	1.7202e−12	3.0198e−11	6.6191e−11	3.0198e−11
	h-value	1	1	1	1	1	1	1	1	1	1	1	1
	z-value	−6.6455	−6.6455	−6.6455	−6.6455	−3.1564	−6.6308	−6.6455	−6.6462	−7.0554	−6.6455	−6.5290	−6.6455
HEPSO	*P*-value	3.0198e−11	3.0198e−11	3.0198e−11	3.0198e−11	3.0198e−11	3.0198e−11	3.0198e−11	3.0066e−11	1.7202e−12	3.0198e−11	1.5955e−11	3.0198e−11
	h-value	1	1	1	1	1	1	1	1	1	1	1	1
	z-value	−6.6455	−6.6455	−6.6455	−6.6455	−6.6455	−6.6455	−6.6455	6.6462	−7.0554	−6.6455	−6.7389	−6.6455
RPSOLF	*P*-value	2.9822e−11	3.0198e−11	1.1019e−11	1.2117e−12	3.0198e−11	3.0198e−11	8.1013e−10	3.0066e−11	0.3337	3.1506e−12	0.0001	3.0198e−11
	h-value	1	1	1	1	1	1	1	1	0	1	1	1
	z-value	6.6474	6.6455	6.7925	7.1040	−6.6455	−6.6455	−6.1429	−6.6462	0.9666	6.9708	3.8008	−6.6455
LFPSO	*P*-value	3.0198e−11	3.0198e−11	3.0198e−11	3.0122e−11	5.5726e−10	3.0198e−11	3.0198e−11	3.0066e−11	1.7202e−12	3.0198e−11	3.1916e−10	3.0198e−11
	h-value	1	1	1	1	1	1	1	1	1	1	1	1
	z-value	−6.6455	−6.6455	−6.6455	6.6459	−6.2020	−6.6455	−6.6455	−6.6462	−7.0554	−6.6455	−6.2891	−6.6455

PSOSCALF Vs	Wilcoxon's rank sum test	F13	F14	F15	F16	F17	F18	F19	F20	F21	F22	F23
PSO	*P*-value	3.1572e−05	2.6908e−06	0.2398	1.3219e−10	N/A	2.4759e−09	2.4650e−09	0.0687	3.7706e−06	0.0023	0.7508
	h-value	1	1	0	1	0	1	1	0	1	1	0
	z-value	−4.1618	−4.6931	−1.1753	6.4246	N/A	5.9630	5.9637	1.8202	−4.6236	−3.0360	0.3174
HPSO-TVAC	*P*-value	1.1023e−08	0.3661	9.7917e−05	0.0005	0.0814	0.0608	0.5290	0.8135	0.1039	0.2275	0.4548
	h-value	1	0	1	1	0	0	0	0	0	0	0
	z-value	−5.7141	0.9037	−3.8956	3.4802	−1.7425	1.8742	0.6294	0.2358	−1.6259	−1.2067	−0.7473
TACPSO	*P*-value	3.3241e−06	0.0027	7.2102e−06	0.0005	N/A	1.7783e−07	2.9971e−07	0.1150	0.5655	0.7668	0.5519
	h-value	1	1	1	1	0	1	1	0	0	0	0
	z-value	−4.6497	2.9922	4.4873	3.4802	N/A	5.2211	5.1236	1.5758	−0.5746	−0.2964	0.5948
MPSO	*P*-value	3.3241e−06	1.8365e−09	0.0042	3.4572e−06	N/A	3.8909e−08	2.9971e−07	0.0386	0.3043	0.7871	0.0001
	h-value	1	1	1	1	0	1	1	1	0	0	1
	z-value	−4.6497	6.0116	2.8607	4.6415	N/A	5.4957	5.1236	2.0681	−1.0270	0.2700	3.8616

(*continued on next page*)

Table C.1 (*continued*).

PSOSCALF Vs	Wilcoxon's rank sum test	F13	F14	F15	F16	F17	F18	F19	F20	F21	F22	F23
AGPSO	*P*-value	1.8608e−06	0.0100	9.8712e−07	2.9971e−07	N/A	2.3625e−08	1.1804e−06	0.1150	0.5775	0.8551	0.4719
	h-value	1	1	1	1	0	1	1	0	0	0	0
	z-value	−4.7679	2.5728	4.8941	5.1236	N/A	5.5831	4.8588	1.5758	−0.5570	0.1825	0.7192
HEPSO	*P*-value	3.0198e−11	6.6764e−07	2.1947e−08	0.0005	0.3337	3.1920e−10	3.0153e−08	0.8287	5.2000e−12	5.1812e−12	1.0840e−11
	h-value	1	1	1	1	0	1	1	0	1	1	1
	z-value	−6.6455	4.9705	−5.5959	−3.4629	−0.9666	−6.2891	−5.5405	0.2163	−6.9000	−6.9005	6.7948
RPSOLF	*P*-value	3.0198e−11	1.0795e−11	3.9647e−08	4.0805e−12	1.2117e−12	2.9045e−11	6.3187e−12	2.0925e−09	5.2000e−12	5.1812e−12	1.0840e−11
	h-value	1	1	1	1	1	1	1	1	1	1	1
	z-value	−6.6455	−6.7954	−5.4924	−6.9343	−7.1040	−6.6513	−6.8722	−5.9904	−6.9000	−6.9005	−6.7948
LFPSO	*P*-value	8.1974e−07	3.0737e−10	1.2017e−08	2.9971e−07	N/A	1.3717e−08	6.8404e−08	0.0908	0.9544	0.4246	0.2173
	h-value	1	1	1	1	0	1	1	0	0	0	0
	z-value	−4.9306	6.2950	−5.6994	5.1236	N/A	5.6768	5.3953	1.6908	−0.0571	0.7984	1.2337

Table C.2
Comparison of PSOSCALF with other algorithms using Wilcoxon's rank sum test.

PSOSCALF Vs	Wilcoxon's rank sum test	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12
GA	*P*-value	3.0198e−11	3.0198e−11	3.0198e−11	3.0198e−11	3.0198e−11	3.0198e−11	3.0198e−11	3.0066e−11	1.7202e−12	3.0198e−11	1.5955e−11	3.0198e−11
	h-value	1	1	1	1	1	1	1	1	1	1	1	1
	z-value	−6.6455	−6.6455	−6.6455	−6.6455	−6.6455	−6.6455	−6.6455	−6.6462	−7.0554	−6.6455	−6.7389	−6.6455
WOA	*P*-value	3.0198e−11	3.0198e−11	3.0198e−11	3.0198e−11	5.5726e−10	3.0198e−11	1.5580e−08	5.4706e−11	1	1.7847e−11	0.0060	3.0198e−11
	h-value	1	1	1	1	1	1	1	1	0	1	1	1
	z-value	6.6455	6.6455	−6.6455	−6.6455	−6.2020	−6.6455	−5.6550	−6.5575	0	6.7226	2.7439	−6.6455
DA	*P*-value	3.0198e−11	3.0198e−11	3.0198e−11	3.0198e−11	3.0198e−11	3.0198e−11	3.0198e−11	3.0066e−11	1.7202e−12	2.8646e−11	1.5955e−11	3.0198e−11
	h-value	1	1	1	1	1	1	1	1	1	1	1	1
	z-value	−6.6455	−6.6455	−6.6455	−6.6455	−6.6455	−6.6455	−6.6455	−6.6462	−7.0554	−6.6533	−6.7389	−6.6455
SCA	*P*-value	3.0198e−11	3.0198e−11	3.0198e−11	3.0198e−11	3.0198e−11	3.0198e−11	3.0198e−11	3.0066e−11	1.7202e−12	3.0198e−11	1.5955e−11	3.0198e−11
	h-value	1	1	1	1	1	1	1	1	1	1	1	1
	z-value	−6.6455	−6.6455	−6.6455	−6.6455	−6.6455	−6.6455	−6.6455	−6.6462	−7.0554	−6.6455	−6.7389	−6.6455
ALO	*P*-value	3.0198e−11	3.0198e−11	3.0198e−11	3.0198e−11	3.0198e−11	3.0198e−11	3.0198e−11	1.6104e−11	1.7202e−12	3.0198e−11	6.6191e−11	3.0198e−11
	h-value	1	1	1	1	1	1	1	1	1	1	1	1
	z-value	−6.6455	−6.6455	−6.6455	−6.6455	−6.6455	−6.6455	−6.6455	−6.7375	−7.055	−6.6455	−6.5290	−6.6455
FA	*P*-value	3.0198e−11	3.0198e−11	3.0198e−11	3.0198e−11	3.0198e−11	3.0198e−11	3.0198e−11	3.0066e−11	1.7202e−12	3.0198e−11	1.5955e−11	3.0198e−11
	h-value	1	1	1	1	1	1	1	1	1	1	1	1
	z-value	−6.6455	−6.6455	−6.6455	−6.6455	−6.6455	−6.6455	−6.6455	−6.6462	−7.0554	−6.6455	−6.7389	−6.6455
MFO	*P*-value	3.0198e−11	3.0198e−11	3.0198e−11	3.0198e−11	3.0198e−11	3.0198e−11	3.0198e−11	3.0066e−11	1.7202e−12	3.0198e−11	1.5955e−11	3.0198e−11
	h-value	1	1	1	1	1	1	1	1	1	1	1	1
	z-value	−6.6455	−6.6455	−6.6455	−6.6455	−6.6455	−6.6455	−6.6455	−6.6462	−7.0554	−6.6455	−6.7389	−6.6455
MVO	*P*-value	3.0198e−11	3.0198e−11	3.0198e−11	3.0198e−11	3.0198e−11	3.0198e−11	3.0198e−11	3.0066e−11	1.7202e−12	3.0198e−11	1.5955e−11	3.0198e−11
	h-value	1	1	1	1	1	1	1	1	1	1	1	1
	z-value	−6.6455	−6.6455	−6.6455	−6.6455	−6.6455	−6.6455	−6.6455	−6.6462	−7.0554	−6.6455	−6.7389	−6.6455

PSOSCALF Vs	Wilcoxon's rank sum test	F13	F14	F15	F16	F17	F18	F19	F20	F21	F22	F23
GA	*P*-value	2.8715e−10	0.0026	2.1543e−10	0.0071	2.2118e−06	2.2225e−08	6.3187e−12	0.3746	5.2000e−12	5.1812e−12	1.0840e−11
	h-value	1	1	1	1	1	1	1	0	1	1	1
	z-value	−6.3055	−3.0012	−6.3499	−2.6901	−4.7330	−5.5937	−6.8722	−0.8877	−6.9000	−6.9005	−6.7948
WOA	*P*-value	3.0198e−11	1.0795e−11	9.2602e−09	4.0805e−12	1.2117e−12	2.9045e−11	6.3187e−12	0.001	5.2000e−12	5.1812e−12	1.0840e−11
	h-value	1	1	1	1	1	1	1	1	1	1	1
	z-value	−6.6455	−6.7954	−5.7437	−6.9343	−7.1040	−6.6513	−6.8722	−3.0959	−6.9000	−6.9005	−6.7948
DA	*P*-value	3.0198e−11	0.9791	1.1957e−10	0.8859	0.0815	0.3318	6.3187e−12	2.0468e−05	8.5167e−11	1.5129e−08	3.9805e−07
	h-value	1	0	1	0	0	0	1	1	1	1	1
	z-value	−6.6455	0.0261	−6.4398	0.1434	−1.7419	0.9703	−6.8722	−4.2597	−6.4911	−5.6600	−5.0698
SCA	*P*-value	3.0198e−11	1.0795e−11	1.1737e−09	4.0805e−12	1.2117e−12	2.9045e−11	6.3187e−12	1.9917e−11	5.2000e−12	5.1812e−12	1.0840e−11
	h-value	1	1	1	1	1	1	1	1	1	1	1
	z-value	−6.6455	−6.7954	−6.0837	−6.9343	−7.1040	−6.6513	−6.8722	−6.7066	−6.9000	−6.9005	−6.7948
ALO	*P*-value	1.3288e−10	1.7326e−10	2.9215e−09	4.0805e−12	1.6666e−08	2.9045e−11	6.3187e−12	0.0037	5.2000e−12	5.1812e−12	1.0840e−11
	h-value	1	1	1	1	1	1	1	1	1	1	1
	z-value	−6.4238	−6.3833	−5.9359	−6.9343	−5.6434	−6.6513	−6.8722	−2.9019	−6.9000	−6.9005	−6.7948

(*continued on next page*)

Table C.2 (*continued*).

PSOSCALF Vs	Wilcoxon's rank sum test	F13	F14	F15	F16	F17	F18	F19	F20	F21	F22	F23
FA	*P*-value	3.0198e−11	1.0788e−11	0.0107	4.0805e−12	1.2117e−12	2.9045e−11	6.3187e−12	0.0074	5.2000e−12	5.1812e−12	1.0840e−11
	h-value	1	1	1	1	1	1	1	1	1	1	1
	z-value	−6.6455	−6.7955	−2.5503	−6.9343	−7.1040	−6.6513	−6.8722	−2.6781	−6.9000	−6.9005	−6.7948
MFO	*P*-value	3.0198e−11	0.0021	3.0469e−09	1.9682e−11	N/A	2.7508e−09	3.7783e−10	0.2589	0.9064	0.1860	0.2181
	h-value	1	1	1	1	0	1	1	0	0	0	0
	z-value	−6.6455	3.0750	−5.9290	6.7083	N/A	5.9458	6.2629	−1.1288	−0.1174	1.3223	1.2315
MVO	*P*-value	3.0198e−11	1.0795e−11	2.4386e−09	4.0805e−12	1.2117e−12	2.9045e−11	6.3187e−12	0.0001	5.2000e−12	5.1812e−12	1.0840e−11
	h-value	1	1	1	1	1	1	1	1	1	1	1
	z-value	−6.6455	−6.7954	−5.9655	−6.9343	−7.1040	−6.6513	−6.8722	−3.7971	−6.9000	−6.9005	−6.7948

References

[1] R.C. Eberhart, Y. Shi, J. Kennedy, Swarm Intelligence, first ed., Morgan Kaufmann, 2001.
[2] X.-S. Yang, Engineering Optimization an Introduction with Metaheuristic Applications, first ed., John Wiley & Sons, New Jersey, 2010.
[3] X.S. Yang, Nature-Inspired Metaheuristic Algorithms, second ed., Luniver Press, 2010.
[4] D. Karaboga, An Idea Based on Honey Bee Swarm For Numerical Optimization, Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.
[5] M. Dorigo, G.D. Caro, Ant colony optimization: a new meta-heuristic, in: Proceedings of the 1999 Congress on Evolutionary Computation, Washington, DC, 1999, pp. 1470–1477.
[6] S.A. Mirjalili, A. Lewis, The whale optimization algorithm, Adv. Eng. Softw. 95 (2016) 51–67.
[7] A.H. Gandomi, X.-S. Yang, A.H. Alavi, Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems, Eng. Comput. 29 (2013) 17–35.
[8] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 1995, pp. 39–43.
[9] M. Li, D. Lin, J. Kou, A hybrid niching PSO enhanced with recombination-replacement crowding strategy for multimodal function optimization, Appl. Soft. Comput. 12 (2012) 975–987.
[10] G. Venter, J. Sobieszczanski-Sobieski, Multidisciplinary optimization of a transport aircraft wing using particle swarm optimization, Struct. Multidiscip. Optim. 26 (2004) 121–131.
[11] D.J. Bordoloi, R. Tiwari, Optimization of SVM methodology for multiple fault taxonomy of rolling bearings from acceleration records, in: 9th IFToMM International Conference on Rotor Dynamics Mechanisms and Machine Science, vol. 21, 2015, pp. 533–542.
[12] X. Zhang, J. Li, J. Xing, P. Wang, Q. Yang, C. He, A particle swarm optimization technique-based parametric wavelet thresholding function for signal denoising, Cricuits Systems Signal Process. 36 (2017) 247–269.
[13] A. Belkadi, B. Daachi, On the robust PID adaptive controller for exoskeletons: A particle swarm optimization based approach, Appl. Soft. Comput. 60 (2017) 87–100.
[14] B. Luitel, G. k. Venayagamoorthy, Particle swarm optimization with quantum infusion for system identification, Eng. Appl. Artif. Intell. 23 (2010) 635–649.
[15] Y. Shi, R.C. Eberhart, Empirical study of particle swarm optimization, in: Proc. IEEE Int. Congr. Evolutionary Computation, vol. 3, 1999, pp. 101–106.
[16] A. Ratnaweera, S.K. Halgamuge, H.C. Watson, Self-Organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients, IEEE Trans. Evol. Comput. 8 (3) (2004) 240–255.
[17] T. Zjyu, Z. Dingxue, A Modified particle swarm optimization with an adaptive acceleration coefficients, in: Asia – Pacific Conference on Information Processing, 2009, pp. 330–332.
[18] G.Q. Bao, K.F. Mao, Particle swarm optimization algorithm with asymmetric time varying acceleration coefficients, in: IEEE International Conference on Robotics and Biomimetics, Guilin, 2009, 2134–2139.
[19] S.A. Mirjalili, A. Lewis, A.S. Sadiq, Autonomous particles groups for particle swarm optimization, Arab. J. Sci. Eng. 39 (2014) 4683–4697.
[20] Z. Cui, J. Zeng, Y. Yin, An improved PSO with time–varying accelerator coefficient, in: Eighth International Conference on Intelligent Systems Design and Applications, 2008, pp. 638–643.
[21] M.J. Mahmoodabadi, A. Bagheri, N. Nariman-zadeh, A. Jamali, A new optimization algorithm based on a combination of particle swarm optimization, convergence and divergence operators for single-objective and multi-objective problems, Eng. Optim. 44 (2012) 1167–1186.
[22] A. Mortazavi, V. Togan, A. Nuhoglu, An integrated particle swarm optimizer for optimization of truss structures with discrete variables, Struct. Eng. Mech. 61 (2017) 359–370.
[23] B. Shuang, J. Chen, Z. Li, Study on hybrid PS-ACO algorithm, Appl. Intell. 34 (2011) 64–73.
[24] K. Premalatha, A.M. Natarajan, Hybrid PSO and GA for global maximization, Int. J. Open. Probl. Comput. Sci. Math. 2 (2009) 597–608.
[25] H. Gray, A hybrid PSO-GA Algorithm for constrained optimization problems, Appl. Math. Comput. 274 (2016) 292–305.
[26] S.A. Mirjalili, S.Z. Mohd Hashim, A new hybrid PSOGSA algorithm for function optimization, in: International Conference on Computer and Information Application, ICCIA 2010, vol. 377, 2010, pp. 374–377.
[27] K.S. Sree Ranjini, S. Murugan, Memory based hybrid dragonfly algorithm for numerical optimization problems, Expert Syst. Appl. 83 (2017) 63–78.
[28] M.J. Mahmoodabadi, Z. Salahshoor Mottaghi, A. Bagheri, HEPSO: high exploration particle swarm optimization, J. Inf. Sci. 273 (2014) 101–111.
[29] A.A. Al-Temeemy, J.W. Spencer, J.F. Ralph, Levy flights for improved ladder scanning, in: 2010 IEEE International Conference on Imaging Systems and Techniques, IST, Thessaloniki, Greece, 2010, pp. 225–228.
[30] S.A. Mirjalili, Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, multi-objective problems, Neural Comput. Appl. 27 (2016) 1053–1073.
[31] S. Amirsadri, S.J. Mousavirad, H. Ebrahimpour-Komleh, A levy flight-based grey wolf optimizer combined with back-propagation algorithm for neural network training, Neural Comput. Appl. (2017),http://dx.doi.org/10.1007/s00521-017-2952-5.
[32] G. Fileccia Scimemi, T. Turetta, C. Celauro, Back calculation of airport pavement moduli and thickness using the Lévy ant colony optimization algorithm, Constr. Build. Mater. 119 (2016) 288–295.
[33] W.A. Hussein, S. Sahran, S.N.H. Sheikh Abdullah, Patch-levy-based initialization algorithm for bees algorithm, Appl. Soft. Comput. 23 (2014) 104–121.
[34] B. Yan, Z. Zhao, Y. Zhou, W. Yuan, J. Li, J. Wu, D. Cheng, A particle swarm optimization algorithm with random learning mechanism and levy flight for optimization of atomic clusters, Comput. Phys. Comm. 219 (2017) 79–86.
[35] H.H.H. Uguz, A novel particle swarm optimization algorithm with levy flight, Appl. Soft. Comput. 23 (2014) 333–345.
[36] R. Jensi, G. Wiselin Jiji, An enhanced particle swarm optimization with levy flight for global optimization, Appl. Soft. Comput. 43 (2016) 248–261.
[37] S.A. Mirjalili, SCA: a sine cosine algorithm for solving optimization problems, Knowl.-Based Syst. 96 (2016) 120–133.
[38] Y. Shi, R. Eberhart, A modified particle swarm optimizer, in: The 1998 IEEE International Conference on Evolutionary Computation Proceedings, Anchorage, AK, 1998, pp. 69–73.
[39] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. Evol. Comput. (2006) 281–295.
[40] X. Zhang, J.L. Li, J.C. Xing, P.Wang, Q.L.Yang, R.H.Wang, C. He, Optimal sensor placement latticed shell structure based on improved particle swarm optimization algorithm, Math. Probl. Eng. (2014),http://dx.doi.org/10.1155/2014/743904.
[41] S.A. Mirjalili, The ant lion optimizer, Adv. Eng. Softw. 83 (2015) 80–98.
[42] X.-S. Yang, Firefly algorithm, stochastic test functions and design optimization, Int. J. Bio-Inspired Comput. 2 (2010) 78–84.
[43] S.A. Mirjalili, Moth-Flame optimization algorithm: a novel nature-inspired heuristic paradigm, Knowl.-Based Syst. 89 (2015) 228–249.
[44] S.A. Mirjalili, S.M. Mirjalili, A. Hatamlou, Multi -verse optimizer: a nature inspired algorithm for global optimization, Neural. Comput. Applic. 27 (2016) 495–513.
[45] C.A. Coello 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 (2002) 1245–1287.
[46] A.H. Gandomi, Interior search algorithm (ISA): A novel approach for global optimization, ISA Trans. 53 (2014) 1168–1183.
[47] E. Sandgren, Nonlinear integer and discrete programming in mechanical design optimization, J. Mech. Des. 112 (1990) 223–229.
[48] A. Sadollah, A. Bahreininejad, H. Eskandar, M. Hamdi, Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems, Appl. Soft. Comput. 13 (2013) 2592–2612.
[49] S.-J. Wu, P.-T. Chow, Genetic algorithms for nonlinear mixed discrete-integer optimization problems via meta-genetic parameter optimization, Eng. Optim+A35 24 (1995) 137–159.
[50] K. Deb, M. Goyal, A combined genetic adaptive search (GeneAS) for engineering design, Comput. Sci. Inform. 26 (1996) 30–45.
[51] T.K. Sharma, M. Pant, V. Singh, Improved Local Search in Artificial Bee Colony Using Golden Section Search, 2012, arXiv preprint arXiv:1210.6128.
[52] B. Kannan, S.K. Kramer, An augmented lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design, J. Mech. Des. 116 (1994) 405–511.
[53] S.A. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer, Adv. Eng. Softw. 69 (2014) 46–61.
[54] M. Mahdavi, M. Fesanghary, E. Damangir, An improved harmony search algorithm for solving optimization problems, Appl. Math. Comput. 188 (2007) 1567–1579.
[55] Q. He, L. Wang, An effective co-evolutionary particle swarm optimization for constrained engineering design problems, Eng. Appl. Artif. Intell. 20 (2007) 20: 89–99.
[56] C.A. Coello Coello, Use of a self-adaptive penalty approach for engineering optimization problems, Comput. Ind. 41 (2000) 113–127.
[57] C.A. Coello Coello, E. Mezura Montes, Constraint-handling in genetic algorithms through the use of dominance-based tournament selection, Adv. Eng. Inf. 16 (2002) 193–203.
[58] K. Deb, GeneAS: a robust optimal design technique for mechanical component design, in: D. Dasgupta, Z. Michalewicz (Eds.), Evolutionary Algorithms in Engineering Applications, Springer, Berlin, Heidelberg, 1997, pp. 497–514.
[59] E. Mezura-Montes, C.A. Coello Coello, An empirical study about the usefulness of evolution strategies to solve constrained optimization problems, Int. J. Gen. Syst. 37 (2008) 443–473.
[60] L. Li, Z. Huang, F. Liu, Q. Wu, A heuristic particle swarm optimizer for optimization of pin connected structures, Comput. Struct. 85 (2007) 340–349.

[61] A. Kaveh, S. Talatahari, An improved ant colony optimization for constrained engineering design problems, Eng. Comput. Int. J. Comput.-Aided Eng. 27 (2010) 155–182.
[62] A. Kaveh, M. M. Khayatazad, A new meta-heuristic method: ray optimization, Comput. Struct. 112 (2012) 283–294.
[63] A.D. Belegundu, Study of mathematical programming methods for structural optimization. Part II: Numerical results, Internat. J. Numer. Methods. Engrg. 21 (1985) 1601–1623.
[64] J.S. Arora, Introduction to Optimum Design, Academic Press, 2004.
[65] G.G. Wang, Adaptive response surface method using inherited latin hypercube design points, J. Mech. Des. 125 (2003) 210–220.
[66] M.-Y. Cheng, D. Prayogo, Symbiotic organisms search: a new metaheuristic optimization algorithm, Comput. Struct. 139 (2014) 98–112.
[67] J.K. Kuang, S.S. Rao, L. Chen, Taguchi-aided search method for design optimization of engineering systems, Eng. Optim. 30 (1998) 1–23.
[68] S. Akhtar, K. Tai, T. Ray, A socio-behavioural simulation model for engineering design optimization, Eng. Optim. 34 (2002) 341–354.
[69] T. Ray, P. Saini, Engineering design optimization using a swarm with an intelligent information sharing among individuals, Eng. Optim. 33 (2001) 735–748.
[70] E.M. Montes, C.A.C. Coello, L. Ricardo, Engineering optimization using a simple evolutionary algorithm, in: 15th Intl. Conf. on Tools with Art. Intelligence—ICTAI'2003, CA, USA, 2003, pp. 149–156.
[71] M. Zhang, W. Luo, X. Wang, Differential evolution with dynamic stochastic selection for constrained optimization, Inform. Sci. 178 (2008) 3043–3074.
[72] J.-F. Tsai, Global optimization of nonlinear fractional programming problems in engineering design, Eng. Optim. 37 (2005) 399–409.
[73] H. Liu, Z. Cai, Y. Wang, Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization, Appl. Soft. Comput. 10 (2010) 629–640.
[74] A. Kaveh, V. Mahdavi, Colliding bodies optimization: a novel meta-heuristic method, Comput. Struct. 139 (2014) 18–27.
[75] A. Carlos, C. Coello, Constraint-Handling using an evolutionary multiobjective optimization technique, Civil. Eng. Syst. 17 (2000) 319–346.
[76] K. Deb, Optimal design of a welded beam via genetic algorithms, AIAA J. 29 (1991) 2013–2015.
[77] K. Deb, An efficient constraint handling method for genetic algorithms, Comput. Methods Appl. Mech. Eng. 186 (2000) 311–338.
[78] K.S. Lee, Z.W. Geem, A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice, Comput. Methods Appl. Mech. Eng. 194 (2005) 3902–3933.
[79] K. Ragsdell, D. Phillips, Optimal design of a class of welded structures using geometric programming, ASME J. Eng. Ind. 98 (1976) 1021–1025.
[80] R.A. Krohling, L. dos Santos Coelho, Coevolutionary particle swarm optimization using gaussian distribution for solving constrained optimization problems, IEEE Trans. Syst. Man Cybern. B 36 (2006) 1407–1416.
[81] M. Nikfar, A. Ashrafizadeh, P. Mayeli, Inverse shape design via a new physical-based iterative solution strategy, Inverse Probl. Sci. Eng. 23 (2015) 1138–1162.
[82] P. Mayeli, M. Nili-Ahmadabadi, H. Besharati-Foumani, Inverse shape design for heat conduction problems via ball spine algorithm, Numer. Heat Transfer B 3 (2016) 249–269.
[83] P. Mayeli, M. Nili-Ahmadabadi, M. Pirzadeh, P. Rahmani, Determination of desired geometry by a novel extension of ball spine algorithm inverse method to conjugate heat transfer problems, Comput. Fluids 154 (2017) 390–406.
[84] H. Hesami, P. Mayeli, Development of the ball-spine algorithm to shape optimization of ducts containing nano-fluid, Numer. Heat Transfer A 70 (2016) 1371–1389.
[85] P. Mayeli, H. Hesami, E. Dolatkhah, Numerical investigation of the MHD forced convection in a straight duct with sinusoidal walls containing water-Al2O3 nanofluid, Numer. Heat Transfer A 71 (2016) 1235–1250.