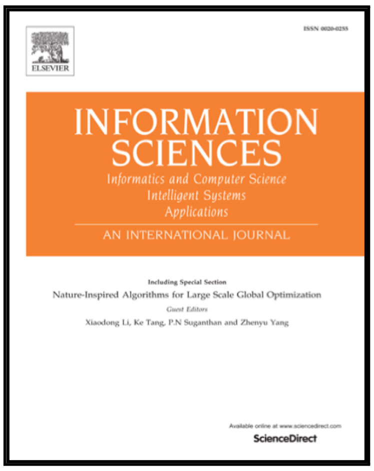

A Hybrid Particle Swarm Optimizer with Sine Cosine Acceleration Coefficients

Ke Chen , Fengyu Zhou , Lei Yin , Shuqian Wang , Yugang Wang , Fang Wan

PII:	S0020-0255(17)30934-9
DOI:	10.1016/j.ins.2017.09.015
Reference:	INS 13119
To appear in:	*Information Sciences*
Received date:	28 April 2017
Revised date:	2 September 2017
Accepted date:	5 September 2017

Highlights

- The sine cosine acceleration coefficients (SCAC) as a new parameter adjustment strategy for the cognitive component c_1 and the social component c_2, respectively.
- The opposition-based learning (OBL) is adopted to initialize population.
- The sine map is utilized to adjust the inertia weight ω.
- Dynamic weight, acceleration coefficient and best-so-far position introduced to update the new position with original update formula.

A Hybrid Particle Swarm Optimizer with Sine Cosine Acceleration Coefficients

Ke Chen Fengyu Zhou* Lei Yin Shuqian Wang Yugang Wang Fang Wan

(School of Control Science and Engineering, Shandong University, Jinan 250061, Shandong, PR China)

*Corresponding author (Fengyu Zhou) fyzhou_sdu@163.com

Abstract: Particle swarm optimization (PSO) has been widely used to solve complex global optimization tasks due to its implementation simplicity and inexpensive computational overhead. However, PSO has premature convergence, is easily trapped in the local optimum solution and is ineffective in balancing exploration and exploitation, especially in complex multi-peak search functions. To overcome the shortcomings of PSO, a hybrid particle swarm optimizer with sine cosine acceleration coefficients (H-PSO-SCAC) is proposed to solve these problems. It is verified by the application of twelve numerical optimization problems. In H-PSO-SCAC, we make the following improvements: First, we introduce sine cosine acceleration coefficients (SCAC) to efficiently control the local search and convergence to the global optimum solution. Second, opposition-based learning (OBL) is adopted to initialize the population. Additionally, we utilize a sine map to adjust the inertia weight ω. Finally, we propose a modified position update formula. Experimental results show that, in the majority of cases, the H-PSO-SCAC approach is capable of efficiently solving numerical optimization tasks and outperforms the existing similar population-based algorithms and PSO variants proposed in recent years. Therefore, the H-PSO-SCAC algorithm is successfully employed as a novel optimization strategy.

Keywords: Particle swarm optimizer; Sine cosine acceleration coefficients; Opposition-based learning; Sine map

1. Introduction

With the increase in the level of industrialization and the development of artificial intelligence technology, stochastic optimization approaches [3, 11] have attracted the attention of technical staff and managers over the past two decades. Optimization refers to the process of searching all the reasonable solutions to determine the optimal solution based on the parameters of a given system to minimize or maximize its output [30]. Optimization tasks are frequently applied in many scientific fields such as engineering design [4], chemistry [14], economics [31], pattern recognition [40] and information theory [48]. In recent years, an increasing number of complex optimization problems have emerged. It is difficult to solve these problems by relying solely on traditional optimization algorithms. Therefore, it is necessary to propose new optimization algorithms. Inspired by the biological and physical phenomena of nature, researchers have proposed a series of intelligent algorithms. Particle swarm optimization (PSO) [23], biogeography-based optimization (BBO) [41], krill herd (KH) algorithm [15], ant colony optimization (ACO) [8], artificial bee colony (ABC) [24], gravitational search algorithm (GSA) [38] and sine cosine algorithm (SCA) [30] are all the well-known paradigms of these intelligence algorithms. These optimization approaches have been adopted by researchers to date and are well suited to solve optimization tasks within various fields such as function optimization [5], feature selection [13], electric transmission systems [18], network attacks [25] and artificial neural networks [26].

The particle swarm optimizer, inspired by the social behaviors of the individuals in flocks of birds, is a nature-inspired and swarm optimization algorithm. Similar to other meta-heuristic swarm intelligent algorithms, PSO begins with the random initialization of population positions in the search range. However, unlike other population-based algorithms, PSO searches for an optimum solution by simply tuning its flying trajectory based on its own best location and its neighborhood's best location at each period [6]. Because of its implementation simplicity and high efficiency, the PSO approach has become a widely accepted optimization method and has been successfully applied to many real-world optimization tasks [7, 9]. The particle swarm optimizer has the disadvantages of lacking diversity and premature convergence. To overcome these problems, variants of PSO have been proposed in the literature, such as orthogonal learning PSO (OLPSO) [49], comprehensive learning PSO (CLPSO) [27], levy flight PSO (LFPSO) [16], time varying acceleration coefficients PSO (TVACPSO) [19] and dynamic multi-swarm PSO (DMS-PSO) [50]. These modified PSO variants have improved search performances over the original PSO. To strengthen the global search performance of PSO and overcome the deficiency of premature convergence, we present a hybrid particle swarm optimizer with sine cosine acceleration coefficients (H-PSO-SCAC) utilizing a new optimization strategy.

In this paper, we propose three modifications to the PSO approach: Initially, we introduce sine cosine acceleration coefficients to avoid premature convergence in the early part of the optimization process and to enhance convergence accuracy at the end of the search stage, which is called PSO-SCAC. The simulation results of twelve numerical functions (seven unimodal and five multimodal functions) show that the PSO-SCAC algorithm has better search accuracy and faster search speed than the PSO and PSO-TVAC methods. Additionally, we apply opposition-based learning (OBL) to initialize the population. Meanwhile, the sine map is used to adjust the inertia weight ω and a new modified formula is proposed to update the next generation population position. These three modifications to PSO will be referred to as H-PSO. Finally, we combine H-PSO with SCAC as a new population search strategy for the PSO concept, which is called H-PSO-SCAC. Again, twelve numerical optimization problems are adopted to verify the performance of H-PSO-SCAC. We establish three group contrast experiments. First, we compare H-PSO-SCAC with H-PSO and the original PSO. Experimental results show that H-PSO-SCAC has better convergence accuracy and a stronger ability to escape local solutions than the original PSO for the majority of test functions, while the H-PSO-SCAC shows better search performance than H-PSO in the majority of cases. Second, the H-PSO-SCAC approach is compared with state-of-art optimization algorithms (ABC, KH, BBO, SCA and GSA). The simulation results show that the H-PSO-SCAC method provides better search results in almost all test functions and it is more stable as well. Third, the H-PSO-SCAC approach is compared with other PSO variants. The results indicate that the H-PSO-SCAC approach outperforms other PSO variants for the majority of numerical functions. In summary, through the above three experiments it can be seen that H-PSO-SCAC has shown a very high search performance on numerical optimization problems. Therefore, the H-PSO-SCAC algorithm should be employed as a novel optimization strategy.

The remainder of this paper is organized as follows: Section 2 summarizes previous related works. In Section 3, we describe the proposed new algorithms (PSO-SCAC, H-PSO and H-PSO-SCAC) that have an extremely strong ability to escape the local optimal solution and have high convergence accuracy. Section 4 describes experimental settings and simulation strategies for numerical function testing. The simulation results and discussions are shown in Section 5. Finally, the paper provides a summary and recommendations for future work in Section 6.

2. Review of previous work

2.1 Particle swarm optimizer

Inspired by the social behaviors of the individuals in flocks of birds, PSO is a nature-inspired and global optimization technique originally developed by Kennedy and Eberhart in the mid-1990s [23]. The PSO algorithm is becoming very popular due to its simplicity, efficiency and ability to quickly and reasonably converge on global optimum solutions. In the simulation, either the best global individual or the best local particle will influence the behavior of each particle in the population. In the original PSO algorithm, a particle represents a potential solution.

When searching in the *D*-dimensional space, each particle *i* has a position vector $\boldsymbol{X_i^d} = [x_{i1}, x_{i2}, \cdots, x_{iD}]$ and a velocity vector $\boldsymbol{V_i^d} = [v_{i1}, v_{i2}, \cdots, v_{iD}]$ to calculate its current state, where *D* is the dimensions of the solution space. Moreover, particle *i* will retain its personal previous best position vector $\boldsymbol{pbest_i} = [pbest_i^1, pbest_i^2, \cdots, pbest_i^D]$. The best position discovered by the entire population is denoted as $\boldsymbol{gbest} = [gbest^1, gbest^2, \cdots, gbest^D]$. The position $\boldsymbol{X_i^d}$ and velocity $\boldsymbol{V_i^d}$ are initialized randomly and updates to the *D*-dimension of the *i* particle. The process are calculated as follows:

$$V_i^d = V_i^d + c_1 \times r_1 \times (pbest_i^d - X_i^d) + c_2 \times r_2 \times (gbest^d - X_i^d) \tag{1}$$

$$X_i^d = X_i^d + V_i^d \tag{2}$$

where c_1 and c_2 are the acceleration parameters which are set to 2.0 commonly and r_1 and r_2 are two uniform distributed values in the range [0,1].

The $V_{\max}$ and $V_{\min}$ parameters may be set to the velocity values determined for each particle to control the velocity value within a reasonable range. In this study, $V_{\max}$ and $V_{\min}$ are set to 20% of the upper and lower values.

The procedure for implementing the PSO can be described briefly by the following steps:

Step 1: Initialization. Initialize the population with random position in the search space and random velocity in the *D*-dimensional problem space.

Step 2: Fitness evaluation. Evaluate the fitness value of each particle based on its position.

Step 3: Compare each particle's fitness value with the particle $pbest$. If the current fitness value is better than the particle $pbest$, then set the $pbest$ value equal to the current value and the $pbest$ location equal to the current location in the search space.

Step 4: Compare the particle's fitness with the previous $gbest$. If the current fitness value is better than $gbest$, then reset $gbest$ to the current particle's value.

Step 5: Based on Eqs. (1) and (2), update the particle's velocity and position, respectively.

Step 6: Return to Step 2 until the maximum number of iterations is reached. The algorithm ends and outputs the optimal solution.

2.2 Background

Since being proposed by Kennedy et al. in 1995, the PSO algorithm has received a high level of attention [7, 49, 50]. One disadvantage of the PSO method is the risk of a premature search convergence. Many researchers contributed to the improvement of the PSO algorithm, thereby deriving many interesting PSO variants. After many numerical simulations, a linear decreasing inertia weight over the course of the search was proposed by Shi and Eberhart [42]. In the original PSO method, the linear decreasing inertia weight is added in Eq. (1) to avoid premature convergence. The modified velocity update mechanism is as follows:

$$V_i^d = \omega \times V_i^d + c_1 \times r_1 \times (pbest_i^d - X_i^d) + c_2 \times r_2 \times (gbest^d - X_i^d) \tag{3}$$

$$\omega = \omega_{\max} - \frac{M_j}{M_{\max}} \times (\omega_{\max} - \omega_{\min}) \tag{4}$$

where M_j and $M_{\max}$ are the current iteration and the maximum iteration, respectively, and $\omega_{\max}$ and

$\omega_{\min}$, defined by the user, are the initial and final weights, respectively. Through empirical studies, Shi and Eberhart have obtained the range of ω as [0.4, 0.9].

Clerc and Kennedy [6] proposed a constriction factor to control or tune the flying velocity. This modification can be represented as follows:

$$V_i^d = \chi \times [V_i^d + c_1 \times r_1 \times (pbest_i^d - X_i^d) + c_2 \times r_2 \times (gbest^d - X_i^d)] \tag{5}$$

where

$$\chi = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|} \tag{6}$$

$$\varphi = c_1 + c_2 \tag{7}$$

Generally, PSO improvement has primarily the following two aspects: First, improving PSO's performance by combining it with other search approaches. Evolutionary operators such as mutation, crossover and selection have been introduced to PSO to enhance its capacity to escape the local optimal values [12], to increase the diversity of its population and to retain the global best particles [1], respectively. Mutation operators are also used to mitigate the premature convergence problem [34]. In Ref. [47], a cooperative approach to particle swarm optimization called CPSO-S$_k$ was proposed and the CPSO-H$_k$ method, combining the original PSO with the CPSO-S$_k$ algorithms, was shown to offer a significant improvement over the original PSO. Inspired by natural evolution, some researchers have introduced speciation [37] and niche approaches [43] into the standard PSO to avoid swarms lacking diversity and to locate as many optimal values as possible. Second, topological structures of the PSO have been widely researched and different types of topologies have been proposed. In Ref. [44], a neighborhood operator was proposed, where the neighborhood of a particle gradually increases until it includes the entire swarm. Liu et al. [28] introduced differential evolution to PSO and proposed a hybrid method called PSO-DE. Based on grouping and reorganization techniques, Zhao and Suganthan [50] proposed dynamic multi-swarm PSO with harmony search to improve PSO's ability to avoid local optima. To overcome premature convergence, Peram et al. [35] developed the fitness-distance-ratio based PSO called FDR-PSO. Mendes et al. [32] proposed a fully informed PSO. The cluster centers are calculated in Ref. [29] to replace the personal best position, its neighbor's best position, or both.

3. Proposed new developments

Although the PSO algorithm is very robust, straightforward and efficient, when compared with other population-based methods, its ability to fine-tune solutions and escape from local optimal are weakened, primarily due to the lack of diversity of its search process [32-39]. Based on an in-depth study of the PSO algorithm, we propose three modified PSO algorithms to enhance its performance. The details are as follows:

3.1 Sine cosine acceleration coefficients (SCAC)

In PSO, c_1 and c_2 are called the cognitive component and the social component, respectively. They are the stochastic acceleration coefficients responsible for modifying the particle velocity side with $pbest$ and $gbest$. Therefore, these two components are very important for obtaining the optimal solution rapidly and accurately.

In Ref. [46], research indicates that when the cognitive component value is greater than the social component value, it will result in excessive wandering of particles in the search space. In contrast, a relatively high value of social component, compared with the cognitive component, will lead individuals to escape from the global optimal prematurely. In addition, Kennedy and Eberhart suggested setting the cognitive component and the social component values to two to make the mean of both stochastic factors in Eq. (1) equal, so that particles would over fly only half the time of the search. In Ref. [44], Suganthan found that the values of the acceleration coefficients change over time.

Generally, in an evolutionary metaheuristic algorithm, it is desirable that the individuals of populations wander through the entire search space. During the early stage of the optimization process, the global search capability should be enhanced in the search space. Conversely, during the latter stage of the optimization process, the convergence capability toward the global optima should be enhanced around the entire search space.

Based on the above analysis, Ratnaveera et al. [39] proposed time-varying acceleration coefficients (TVAC) to balance the early stage's global search and the latter stage's global converge abilities effectively. When the cognitive component is reduced, the social component will increase. The c_1 and c_2 will be changed over time. PSO based on time-varying acceleration coefficients is called PSO-TVAC. In this algorithm, the time-varying acceleration coefficients can be represented mathematically as follows:

$$c_1 = \left(c_{1f} - c_{1i}\right) \times \frac{M_j}{M_{\max}} + c_{1i} \tag{5}$$

$$c_2 = \left(c_{2f} - c_{2i}\right) \times \frac{M_j}{M_{\max}} + c_{2i} \tag{6}$$

where c_{1f}, c_{1i}, c_{2f} and c_{2i} are constants and M_j and $M_{\max}$ are the current iteration and maximum iteration, respectively. Through simulation experiments, Ratnaveera et al. found that the best range of values for c_1 is between 2.5 and 0.5 and for c_2 is between 0.5 and 2.5.

Considering those concerns, in this paper, we propose sine cosine acceleration coefficients (SCAC) as a new parameter adjustment strategy for the cognitive and social components. Compared with the TVAC, the SCAC can better balance the early stage's global search and the latter stage's global convergence. In SCAC, the range of c_1 is between 2.5 and 0.5 and the range of c_2 is between 0.5 and 2.5. This sine cosine acceleration coefficients improved PSO is called PSO-SCAC. The results of comparing PSO-SCAC with PSO and PSO-TVAC will be presented and discussed in Section 5. The SCAC's trend is shown in Fig. 1.

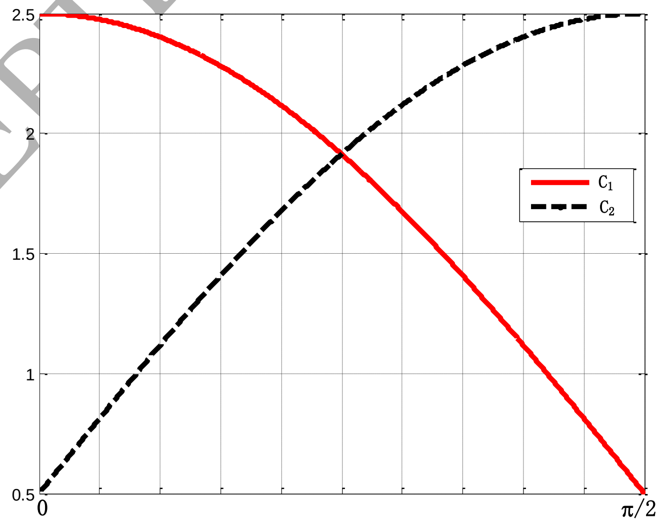

Fig. 1 Sine cosine acceleration coefficients (SCAC)

This modification is described mathematically as follows:

$$c_1 = \partial \times \sin\left(\left(1 - \frac{M_j}{M_{\max}}\right) \times \frac{\pi}{2}\right) + \delta \tag{7}$$

$$c_2 = \partial \times \cos\left(\left(1 - \frac{M_j}{M_{\max}}\right) \times \frac{\pi}{2}\right) + \delta \tag{8}$$

where ∂ and δ are constant ($\partial = 2$, $\rho = 0.5$).

3.2 A hybrid particle swarm optimizer (H-PSO)

The crucial advantage of the PSO algorithm is its simplicity and effectiveness, which makes it easy to implement and lends itself to parallel computation. However, PSO could reach premature convergence and is easily trapped in local optimal solutions. Therefore, a hybrid particle swarm optimizer called H-PSO is proposed to enhance the optimization performance of the original PSO. In the H-PSO method, there are three major modifications, which are expressed in detail as follows:

First, opposition-based learning (OBL) [2] is adopted to initialize the population of the PSO algorithm. We know that the initial population of the PSO method is generated randomly, which may affect the convergence speed and the precision of the final solution. In the absence of any prior knowledge, we use the opposition-based learning instead of the random initial population position as the new initial population strategy. It can increase the opportunities of reaching the global optimal solution by fifty percent. In addition, using the OBL as the initial population strategy can not only improve the quality of the initial solution but also accelerate the global convergence rate. Based on the above analysis, in this paper, we use the OBL to initialize the population. The initialization process is as follows (D denotes the feasible dimension, NP represents the number in the population and M is the current number of iterations):

ⅰ. Random initial population $P(M=0) = \{x_{ij}\}$ $i = 1,2,\cdots,NP$, $j = 1,2,\cdots,D$;

ⅱ. The reverse population $P'(M=0) = \{x'_{ij}\}$ is calculated based on Eq. (9);

$$x'_{ij} = x_{\max,j} + x_{\min,j} - x_{ij} \tag{9}$$

where $x_{\max,j}$ and $x_{\min,j}$ are the population position x_i at j-dimension's max value and min value, respectively.

ⅲ. Select the smaller half of the fitness values from the combined population $\{P(M=0) \cup P'(M=0)\}$ as the initial position of the population.

Second, the sine map [33] is used to adjust the inertia weights ω of the PSO method during the search process. We know from Trelea [46] and Clerc's [6] studies that the parameter ω is an important factor that affects the original PSO algorithm's global convergence. In addition, the PSO's population diversity maintenance is lower, causing premature convergence. The sine map has the properties of ergodicity, non-repetition and irregularity. These behaviors can be analyzed based on the attractor theory and the meaning of the Lyapunov exponents [45]. The sine map used in the PSO method not only can enhance population diversity in the search process but can also strengthen the ability to converge at the global optimal. Therefore, this paper introduces the sine map to tune the inertia weight ω. The range of the sine map is [0,1]. Mathematically, the inertia weight ω is given as follows:

$$\omega = x_k = \frac{c}{4} \times \sin(\pi x_{k-1}) \qquad x_k \in (0,1), \quad 0 < c \le 4, \quad k = 1,2,\cdots,M_{\max} \tag{10}$$

where k is the current iteration number.

Third, a modified position update formula is proposed. It is common knowledge that the abilities of exploration and exploitation contradict. Therefore, to achieving better optimization performance, we must balance the exploration and exploitation abilities effectively. As seen from Eq. (2), the new position obtained depends primarily on two factors: the previous position X_i^d and the velocity V_i^d. To further improve the performances of PSO, we propose three major changes by introducing dynamic weight, acceleration coefficient and best-so-far position to update the new position with these two factors. The search form of the improved PSO, described in Eq. (11), can successfully balance the exploration and exploitation abilities. The operational process can be modified as follows:

$$X_i^d = X_i^d \times w_{ij} + V_i^d \times w'_{ij} + \rho \times gbest^d \times \psi \tag{11}$$

where w_{ij} and w'_{ij} are the dynamic weights that control the influence of the previous solution X_i^d and the velocity V_i^d, respectively; $gbest^d$ is current optimal position that can accelerate the convergence speed; ψ is the acceleration coefficient that determines the maximum step size and ρ is a random number between 0 and 1.

In this paper, dynamic weight and the acceleration coefficient are defined as functions of fitness in the PSO search process. The parameters w_{ij}, w'_{ij} and ψ are defined as follows:

$$w_{ij} = \psi = \frac{\exp(f(j)/u)}{1+\exp(-f(j)/u)^{iter}} \tag{12}$$

$$w'_{ij} = 1 - w_{ij} \tag{13}$$

where u is the mean fitness value in the first iteration, $iter$ is the current iteration and $f(j)$ is the fitness of the *j*th particle.

3.3 A hybrid particle swarm optimizer with sine cosine acceleration coefficients (H-PSO-SCAC)

The PSO algorithm is considered to be a very useful technique in many engineering applications. The essential advantages of the PSO method are that it is easy, simple and can be implemented in any working environment. In recent years, many successive PSO variants have been presented. However, PSO's deficiencies of premature convergence, lack of diversity and being easily trapped in a local optimal solution still exist. To overcome these shortcomings, in this paper, we propose a new population search strategy in which the hybrid particle swarm optimizer (H-PSO) and sine cosine acceleration coefficients (SCAC) are incorporated into the PSO concept. Hereafter, this will be referred to as the H-PSO-SCAC algorithm.

Simulations are performed with classical benchmark functions (seven unimodal and five multimodal problems) to verify the search performance of the H-PSO-SCAC algorithm.

4. Experimental settings and simulation strategies for numerical function testing

4.1 Numerical functions

We choose seven unimodal and five multimodal functions to verify the performance of the proposed methods. In addition, based on their properties, these numerical functions are divided into two groups: unimodal problems and multimodal problems. The equations and properties of these benchmark functions are listed below.

Group A: Unimodal problems:

1) Sphere function

$$f_1(\vec{x}) = \sum_{i=1}^{n} x_i^2 \tag{14}$$

where $x \in [-100, 100]^n$.

2) Schwefel 2.22 function

$$f_2(\vec{x}) = \sum_{i=1}^{n} |x_i| + \prod_{i=1}^{n} |x_i| \tag{15}$$

where $x \in [-10, 10]^n$.

3) Schwefel 1.2 function

$$f_3(\vec{x}) = \sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2 \tag{16}$$

where $x \in [-100, 100]^n$.

4) Schwefel 2.21 function

$$f_4(\vec{x}) = \max_i\{|x_i|, 1 \le i \le n\} \tag{17}$$

where $x \in [-100, 100]^n$.

5) Rosenbrock's function

$$f_5(\vec{x}) = \sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2] \tag{18}$$

where $x \in [-30, 30]^n$.

6) Step function

$$f_6(\vec{x}) = \sum_{i=1}^{n}([x_i + 0.5])^2 \tag{19}$$

where $x \in [-100, 100]^n$.

7) Noise function

$$f_7(\vec{x}) = \sum_{i=1}^{n} ix_i^4 + random[0,1) \tag{20}$$

where $x \in [-1.28, 1.28]^n$.

In this group, among the seven benchmark functions, f_1 is the sphere function and is easily solved, its optimum location is $[0]^n$, global optima is 0 and n denotes its solution space dimension. f_2, f_3 and f_4 are the Schwefel functions, their optimum locations are $[0]^n$ and their global optima are 0. The fifth problem (f_5) is the Rosenbrock function, which may be regarded as a unimodal or multimodal numerical function. It has a narrow gap between the perceived local optima and the global optimum; its optimum location is $[1]^n$ and its global optima is 0. f_6 is the step function, its optimum location is $[1]^n$ and its global optima is 0. The seventh problem (f_7) is the noise function. Because there is a perturbation factor (random), it is difficult to find the global optimal value, but its optimum location is $[1]^n$ and its global optima is 0. Through the seven unimodal problems, we can verify the convergence accuracy and convergence speed of the algorithm.

Group B: Multimodal problems:

8) Rastrigin's function

$$f_8(\vec{x}) = \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i) + 10) \tag{21}$$

where $x \in [-5.12, 5.12]^n$.

9) Ackley's function

$$f_9(\vec{x}) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)) + 20 + e \tag{22}$$

where $x \in [-32, 32]^n$.

10) Griewank's function

$$f_{10}(\vec{x}) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1 \tag{23}$$

where $x \in [-600, 600]^n$.

11) Penalized1 function

$$f_{11}(\vec{x}) = \frac{\pi}{n}\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4) \tag{24}$$

where $y_i = 1 + \frac{x_i + 1}{4}$; $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \le x_i \le a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$, $x \in [-50, 50]^n$.

12) Penalized2 function

$$f_{12}(\vec{x}) = \frac{1}{10}\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2\left[1 + \sin^2(3\pi x_i + 1)\right] + (x_n - 1)^2\left[1 + \sin^2(2\pi x_n)\right]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4) \tag{25}$$

where $y_i = 1 + \frac{x_i + 1}{4}$; $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \le x_i \le a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$, $x \in [-50, 50]^n$.

In this group, there are five multimodal benchmark functions. Rastrigin's function (f_8) has a significant number of local optima. Hence, a method capable of maintaining larger diversity will likely yield better performance. Ackley's function (f_9) has a narrow global basin and a large number of local optima. Griewank's function (f_{10}) has a component that causes linkage among the dimensions; therefore, it will be difficult to reach the global optimum. The penalized functions (f_{11} and f_{12}) are continuous, but they have many minor local optima, thereby making it difficult to determine the global optimum. Through the five multimodal problems, we can verify the ability to escape from the local optimum solution of the algorithm.

4.2 Experimental setting

In this subsection, to verify the performance of the proposed algorithms, the twelve previously mentioned well-known benchmark functions are adopted. A detailed description of the functions was provided in subsection 4.1. The appropriate constraint handling strategies must be chosen to address the constraint problem. Therefore, this paper adopted penalty-based mechanisms in its proposed algorithms [20]. In subsection 4.1, *n* denotes the dimension. The global optimal positions of Rosenbrock's function (f_5) and Step's function (f_6) are $[1]^n$ and $[-0.5]^n$, respectively. The global optimal position of the other ten numerical functions is $[0]^n$. All benchmarks' theoretical optimal values are 0. To obtain an unbiased comparison of CPU times, all the simulation experiments are performed using the same PC, whose configuration is shown in Table 1.

Table 1 The PC Configuration

Name	Detailed settings
Hardware	
CPU	Intel core i5 2410M
Frequency	2.3 GHz
RAM	6 GB
Hard drive	500 GB
Software	
Operating system	Windows 7 (64Bit)
Language	MATLAB R2014a

4.3 Simulation strategies

Simulation experiments are performed to observe the quality of the global optimum solution and the rate of convergence of the new algorithms proposed in this investigation in comparison with PSO-TVAC, other well-known evolutionary methods and some state-of-the-art PSO variants. The classical benchmark functions listed in subsection 4.1 were each tested, with dimensions of 10, 30 and 50. For each optimization function, 20 trials were executed and the mean optimal values and the standard deviations (S.D.) are presented.

In this paper, all empirical experiments were conducted with a population size of 40 and the maximum number of cycle was set to 1000 for all functions. The control parameters of PSO can be obtained from Ref. [21]. In addition, the use of different criteria for different classical benchmark functions is as reported in Ref. [29]. However, all test functions have the global optimum solution of zero. Therefore, for benchmark numerical functions, the maximum number of cycles is established as the stopping criteria.

Furthermore, to determine whether the results obtained by H-PSO-SCAC are significantly different from the results generated by other methods, the nonparametric Wilcoxon rank-sum tests are executed to compare the results of the H-PSO-SCAC and the best result achieved by the other algorithms for each numerical function. The *k* values presented in column 3 of Tables 3 and 5 are the results of the *t*-tests. A *k* value of one indicates that the performances of the two algorithms are significantly different with 95% certainty, whereas a *k* value of zero implies that the performances are not significantly different.

5. Results and discussion

In this section, four groups of simulation experiments are performed to verify the search performance of the H-PSO-SCAC method; the specific steps are described below. In all tables, the best mean of objective values and the best standard deviations of the benchmark numerical functions obtained by the algorithms are shown in bold font.

5.1 Results and comparison of PSO, PSO-TVAC and PSO-SCAC

In this subsection, a series of experiments are presented to show the effectiveness of the sine cosine acceleration coefficients (SCAC). Table 2 shows the execution time (E.T.), the mean results (mean) and the standard deviations (S.D.) of the PSO, PSO-TVAC and PSO-SCAC algorithms. Here, the E.T. represents a single run CPU time. Figs. 2 through 5 show the convergence graphs of the PSO, PSO-TVAC and PSO-SCAC algorithms, respectively.

As seen in Table 2, the solutions of the PSO-SCAC algorithm are better than the PSO and PSO-TVAC algorithms for the majority of functions. For unimodal functions, the PSO-SCAC algorithm has provided better solutions in five functions (f_3, f_4, f_5, f_6 and f_7), except for f_3, f_4, f_5 and f_7 under Dim=10. In two unimodal functions, the PSO-SCAC could not determine a better solution than the PSO-TVAC algorithm. For multimodal functions, the PSO-SCAC method has determined better solutions for all functions (f_8, f_9, f_{10}, f_{11} and f_{12}) except for f_8 and f_{11} under Dim=10. In addition, when comparing the execution times of the three algorithms, we can clearly see that the PSO-SCAC algorithm's execution time is the shortest for all functions, except for f_1 under Dim=10. Until now, there has been no clear theory to explain the use of the sine cosine acceleration coefficients to replace the determinate or time-varying acceleration coefficients to shorten the execution time of PSO algorithms. However, empirical studies indicate that the SCAC strategy can effectively reduce the computational complexity of the PSO algorithm.

Figs. 2 through 5 show the convergence graphs obtained from the PSO, PSO-TVAC and PSO-SCAC algorithms on the four functions (f_3, f_4, f_5 and f_{11}), respectively. The values shown in these figures are the average function optimum achieved from 20 independent experiments. From Fig. 2, PSO demonstrates better search performance than PSO-TVAC and PSO-SCAC. From Figs. 3 through 5, it can clearly be seen that the PSO-SCAC algorithm has better search accuracy and faster search speed than the PSO and PSO-TVAC methods.

Table 2 The means and standard deviations of the PSO-TVAC and PSO-SCAC algorithms

Function	Dim	PSO			PSO-TVAC			PSO-SCAC		
		E.T.	Mean	S.D.	E.T.	Mean	S.D.	E.T.	Mean	S.D.
f_1	10	**0.8736**	6.2012×10^{-10}	6.6764×10^{-10}	0.9906	**3.0884×10^{-10}**	**7.5950×10^{-10}**	0.9828	5.7862×10^{-10}	6.6427×10^{-10}
	30	1.2792	3.7623×10^{-2}	1.3915×10^{-2}	1.0764	**2.2187×10^{-2}**	**8.9205×10^{-3}**	**1.0682**	2.5089×10^{-2}	8.1843×10^{-3}
	50	1.3962	3.8913	7.3184×10^{-1}	1.2090	**2.8482**	**8.1405×10^{-1}**	**1.1462**	3.2666	9.6701×10^{-1}
f_2	10	3.0962	2.1333×10^{-6}	2.0538×10^{-6}	2.7144	**1.8667×10^{-6}**	**2.2492×10^{-6}**	**2.5974**	2.8053×10^{-6}	3.1126×10^{-6}
	30	7.2229	6.0732×10^{-2}	1.1693×10^{-2}	6.2167	**4.3466×10^{-2}**	**9.2468×10^{-3}**	**6.1181**	5.0465×10^{-2}	1.3160×10^{-2}
	50	11.1931	9.9744×10^{-1}	1.4371×10^{-1}	9.9295	**7.5040×10^{-1}**	**9.8186×10^{-2}**	**9.8606**	8.2716×10-1	1.1160×10^{-1}
f_3	10	3.0030	**1.3354×10^{-5}**	**1.6915×10^{-5}**	2.6520	1.8693×10^{-5}	1.6221×10^{-5}	**2.5846**	1.9090×10^{-5}	1.5921×10^{-5}
	30	7.0903	1.1100×10^{2}	4.4931×10^{1}	6.4819	6.5456×10^{1}	3.3255×10^{1}	**6.3726**	**6.4956×10^{1}**	**4.2330×10^{1}**
	50	11.2399	3.2106×10^{3}	9.3806×10^{2}	10.6471	1.8610×10^{3}	5.8595×10^{2}	**10.1243**	**1.6190×10^{3}**	**4.0804×10^{2}**
f_4	10	1.3104	**2.2019×10^{-3}**	**1.6108×10^{-3}**	1.2714	3.2268×10^{-3}	2.2714×10^{-3}	**1.1310**	3.3616×10^{-3}	1.5392×10^{-3}
	30	2.3010	1.4001	2.7135×10^{-1}	2.0202	1.1493	3.1791×10^{-1}	**1.9422**	**1.1216**	**1.7280×10^{-1}**
	50	3.1668	5.6272	9.8911×10^{-1}	2.8938	4.7231	6.2060×10^{-1}	**2.6704**	**4.4259**	**5.7418×10^{-1}**
f_5	10	4.3993	**6.4367×10^{-2}**	**7.0667×10^{-2}**	4.1731	7.2022×10^{-1}	9.4804×10^{-1}	**4.1340**	5.0957×10^{-1}	1.6571
	30	11.8171	1.2644×10^{2}	1.0648×10^{2}	10.8265	8.4452×10^{1}	4.7906×10^{1}	**10.7484**	**7.4741×10^{1}**	**3.7777×101**
	50	19.3441	4.3369×10^{2}	3.1511×10^{2}	18.0805	3.9111×10^{2}	2.3500×10^{2}	**17.9389**	**3.3778×10^{2}**	**1.3772×10^{2}**
f_6	10	1.4664	1.9100×10^{-1}	4.6294×10^{-1}	1.1856	**0**	**0**	**1.1700**	**0**	**0**
	30	2.4258	8.5892×10^{1}	3.4675×10^{1}	2.0358	5.0000×10^{-2}	2.2361×10^{-1}	**2.0114**	**0**	**0**
	50	3.3696	4.5276×10^{2}	3.7128×10^{2}	2.9094	9.7000	2.1300	**2.8684**	**8.8500**	**2.4339**
f_7	10	1.5678	**3.4692×10^{-24}**	**9.1195×10^{-24}**	1.3338	6.7229×10^{-24}	5.9271×10^{-24}	**1.2246**	5.2791×10^{-24}	7.3635×10^{-24}
	30	3.0108	7.3742×10^{-10}	5.6203×10^{-10}	2.2776	4.5803×10^{-10}	2.8601×10^{-10}	**2.3322**	**3.6569×10^{-10}**	**4.5774×10^{-10}**
	50	3.7908	4.3871×10^{-6}	1.8612×10^{-6}	3.2994	1.5456×10^{-6}	7.0869×10^{-7}	**3.2292**	**1.1550×10^{-6}**	**1.1525×10^{-6}**
f_8	10	1.5600	**9.0701×10^{-9}**	**1.7324×10^{-8}**	1.3260	1.2834×10^{-8}	2.8206×10^{-8}	**1.3160**	1.0068×10^{-8}	1.2980×10^{-8}
	30	2.5272	1.1057×10^{1}	2.7928	2.1840	6.9225	3.1842	**2.1431**	**6.6399**	**3.5115**
	50	3.6036	5.7297×10^{1}	1.0915×10^{1}	2.9952	4.1930×10^{1}	7.9372	**2.9576**	**4.1136×10^{1}**	**7.5375**
f_9	10	1.3182	1.2703×10^{-5}	9.4013×10^{-6}	1.1310	6.3111×10^{-6}	3.8228×10^{-6}	**1.1266**	**5.5721×10^{-6}**	**5.9647×10^{-6}**
	30	2.3556	5.9541×10^{-2}	1.6829×10^{-2}	2.0514	5.4995×10^{-2}	1.4605×10^{-2}	**1.9968**	**5.1284×10^{-2}**	**1.3995×10^{-2}**
	50	3.4242	1.2146	3.5069×10^{-1}	2.6832	9.3199×10^{-1}	2.7019×10^{-1}	**2.6288**	**9.1323×10^{-1}**	**3.9277×10^{-1}**
f_{10}	10	1.3338	6.1674×10^{-2}	2.7533×10^{-2}	1.2168	6.1248×10^{-2}	2.5476×10^{-2}	**1.1324**	**5.2034×10^{-2}**	**2.2568×10^{-2}**
	30	2.6676	9.7200×10^{-2}	4.2895×10^{-2}	2.1528	8.2009×10^{-2}	4.2321×10^{-2}	**2.1386**	**8.1617×10^{-2}**	**3.8191×10^{-2}**
	50	3.8610	1.0200	3.8234×10^{-2}	3.4086	9.5009×10^{-1}	6.3260×10^{-2}	**3.2526**	**9.1036×10^{-1}**	**4.5923×10^{-2}**
f_{11}	10	2.5662	**1.4684×10^{-12}**	**2.4656×10^{-12}**	2.1918	6.6249×10^{-12}	2.1086×10^{-11}	**2.0202**	1.4186×10^{-11}	3.6858×10^{-11}
	30	5.5225	1.2987×10^{-4}	8.0601×10^{-5}	4.3681	8.3463×10^{-5}	3.4879×10^{-5}	**4.2836**	**8.1684×10^{-5}**	**6.6072×10^{-5}**
	50	8.8765	1.6937×10^{-1}	3.2479×10^{-1}	6.7549	4.3259×10^{-2}	6.4539×10^{-2}	**6.6328**	**2.7655×10^{-2}**	**3.3493×10^{-2}**
f_{12}	10	2.5506	3.3318×10^{-11}	5.9596×10^{-11}	2.0403	1.9870×10^{-11}	3.7747×10^{-11}	**2.0124**	**1.8355×10^{-11}**	**2.4450×0^{-11}**
	30	5.2807	3.1048×10^{-3}	2.9633×10^{-3}	4.1185	1.8911×10^{-3}	2.5093×10^{-3}	**4.0730**	**1.8264×10^{-3}**	**9.8421×10^{-4}**
	50	8.2681	5.6246×10^{-1}	1.6653×10^{-1}	6.5443	3.5507×10^{-1}	1.5311×10^{-1}	**6.3570**	**3.0765×10^{-1}**	**8.8580×10^{-2}**

E.T.: Execution time; Mean: mean of objective values; S.D.: standard deviation.

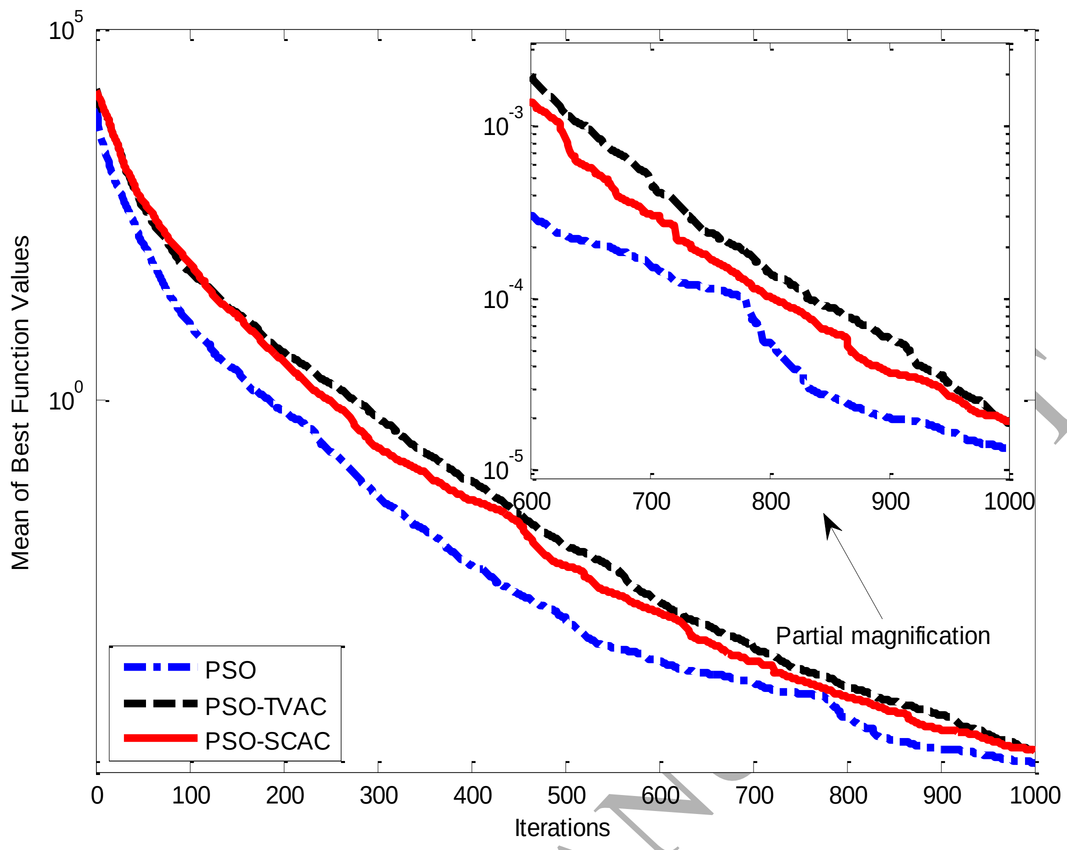

Fig. 2 Comparison of performances of three algorithms for minimization of f_3 with *Dim* = 10

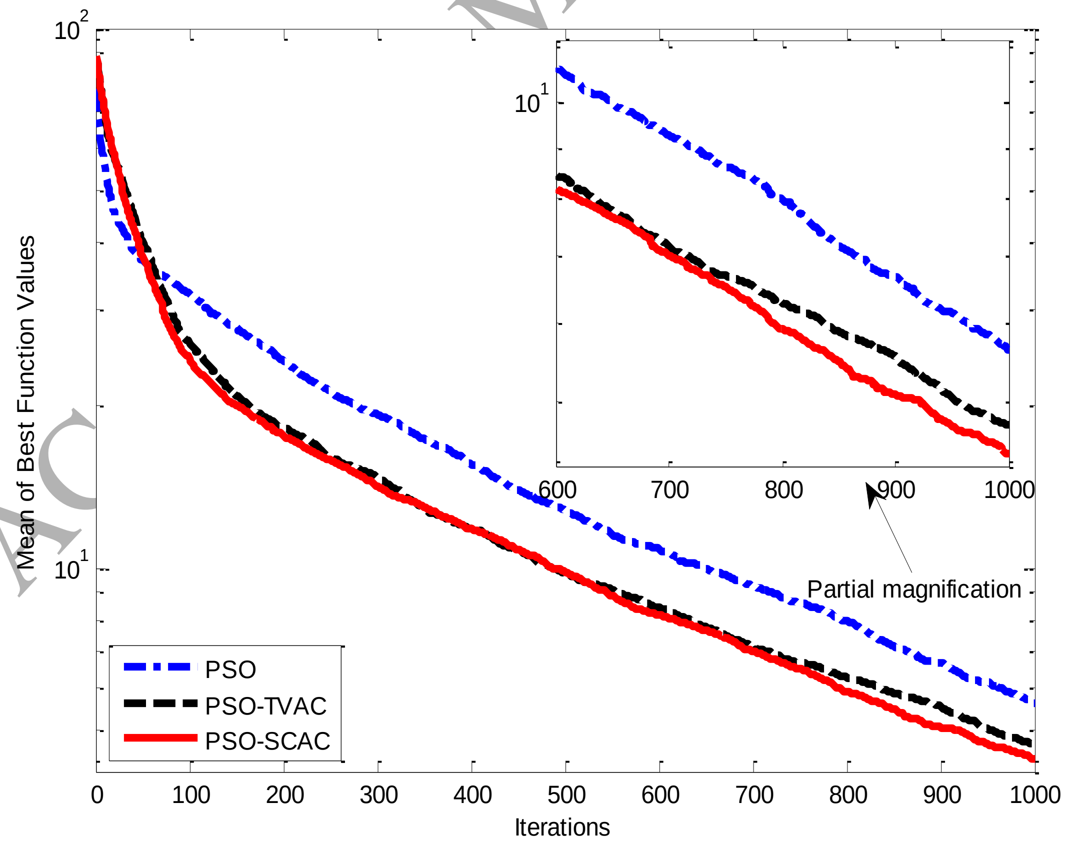

Fig. 3 Comparison of performances of three algorithms for minimization of f_4 with *Dim* = 50

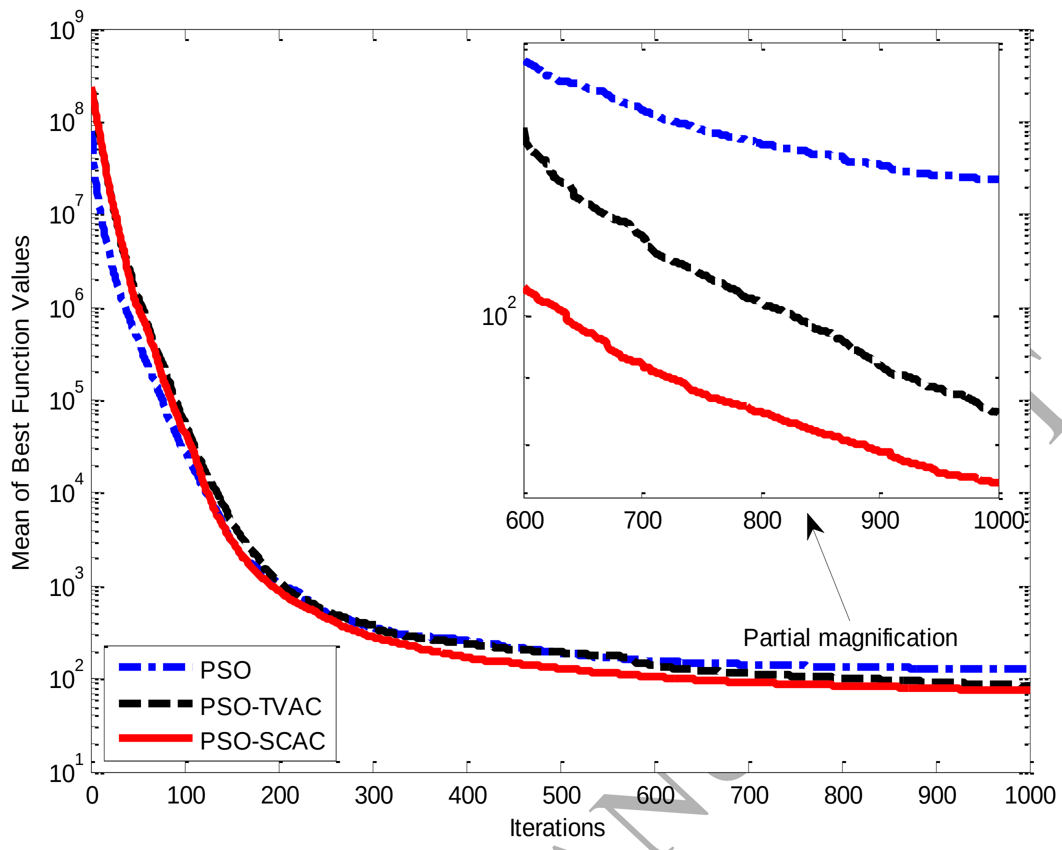

Fig. 4 Comparison of performances of three algorithms for minimization of f_5 with *Dim* = 30

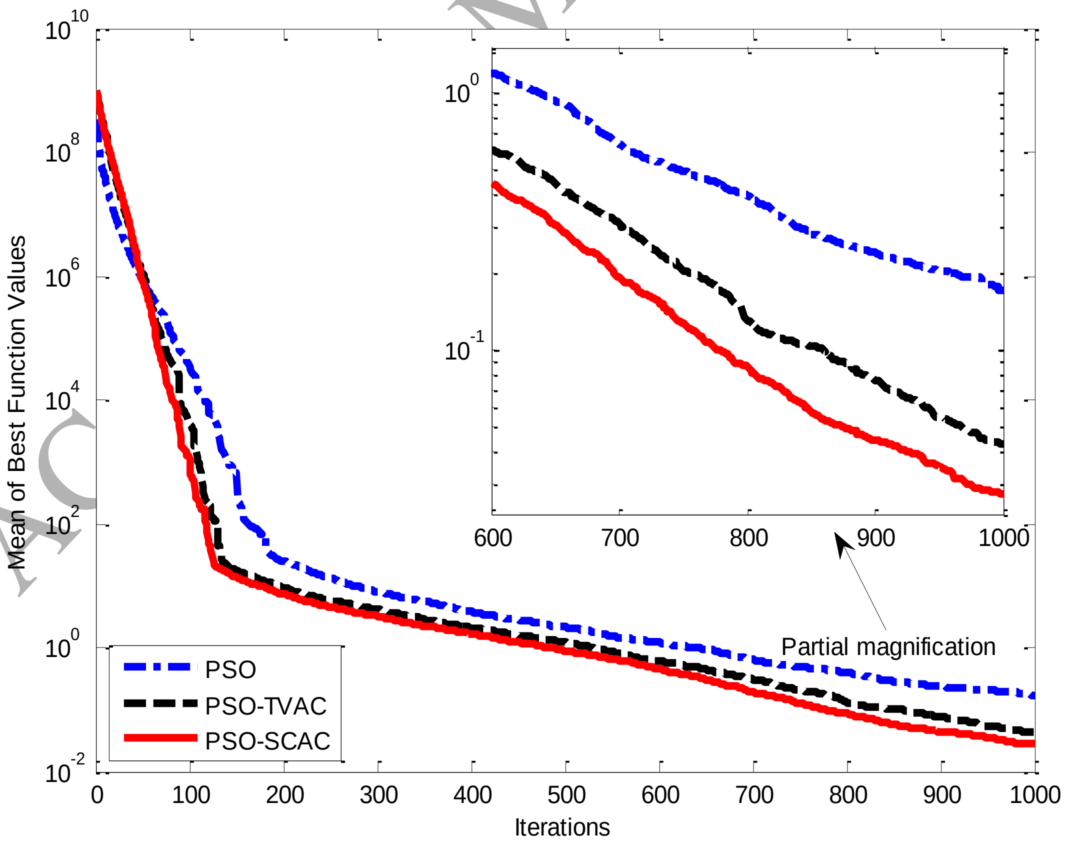

Fig. 5 Comparison of performances of three algorithms for minimization of f_{11} with *Dim* = 50

As seen from Table 2 and Figs. 2 through 5, although PSO-SCAC could not find better solutions than the PSO and PSO-TVAC algorithms for a few functions, the PSO-SCAC algorithm could have a better search accuracy and faster convergence speed than the PSO and PSO-TVAC technology for the majority of functions. Therefore, the proposed SCAC as a new parameter automation strategy is successful for the PSO algorithm.

5.2 Experimental results and comparison of PSO, H-PSO and H-PSO-SCAC

In this subsection, a series of experiments are implemented to show the superiority of the H-PSO-SCAC algorithm in escaping local optimal, convergence speed and search global optimal over H-PSO and the original PSO method. Table 3 shows the convergence iterations, the mean results and the standard deviations of the PSO, H-PSO and H-PSO-SCAC algorithms. Figs. 6 through 11 show the convergence curves of the PSO, H-PSO and H-PSO-SCAC algorithms.

Group A (unimodal functions): For the seven unimodal functions, the results show that the H-PSO-SCAC algorithm outperforms the H-PSO and the traditional PSO methods. As seen from Table 3, the H-PSO-SCAC method is superior to the PSO algorithm on all unimodal functions (f_1, f_2, f_3, f_4, f_5, f_6 and f_7), except for function f_5 under Dim=10. In addition, the H-PSO-SCAC algorithm is superior to H-PSO on functions f_1, f_2, f_3, f_4, f_5, f_6 and f_7, except for function f_2 under Dim=30 and functions f_3, f_5 and f_7 under Dim=10. The H-PSO-SCAC algorithm located the global optimal solutions on two functions (f_4 and f_6), except for function f_4 under *Dim*=10 and Dim=30. For five functions (f_1, f_2, f_3, f_4 and f_7), except for functions f_2 and f_4 under Dim=30, the H-PSO-SCAC algorithm could achieve solutions quite close to the global optimal. However, in only one function (f_5) the H-PSO-SCAC algorithm could not locate superior values within the specified maximum number of iterations. Moreover, analysis of the number of convergence iterations indicates that the H-PSO-SCAC method has faster global convergence for the majority of the unimodal functions.

The plots in Figs. 6 through 8 show the convergence curves of the mean of best function values during the 20 executions for functions f_3, f_6 and f_7, respectively. It is apparent that the H-PSO-SCAC algorithm performs better than the H-PSO method and the traditional PSO in terms of global convergence, escaping from local optimum values and final search solutions. It can be observed from the figures that H-PSO-SCAC with SCAC and H-PSO converge considerably faster than the H-PSO and traditional PSO algorithms.

Group B (multimodal functions): The opposition-based learning (OBL), sine map, modified position update formula and SCAC strategy are added to the PSO algorithm, which can provide abilities to search, preserve and utilize useful information from the learning exemplars. It is expected that the H-PSO-SCAC algorithm can enhance search performance and avoid being trapped in local optimal solutions on multimodal functions. The experimental results of the five multimodal functions (f_8, f_9, f_{10}, f_{11} and f_{12}) are given in Table 3. H-PSO-SCAC surpasses PSO on three functions (f_8, f_9 and f_{10}), achieves the global optimal solution on the f_8 and f_{10} functions and provides solutions quite close to the global optimal value in the f_9 function. However, for two multimodal functions (f_{11} and f_{12}) the H-PSO-SCAC algorithm could not locate better values than PSO under the specified maximum number of iterations. The search performance of the H-PSO-SCAC algorithm is slightly better than H-PSO. In summary, it is shown that the H-PSO-SCAC algorithm is much more effective and robust for multimodal optimization problems.

The convergence of the PSO, H-PSO and H-PSO-SCAC algorithms optimizing the multimodal functions f_8, f_{10} and f_{12}, are plotted in Figs. 9 through 11, respectively. From Fig. 9, it can be observed that the H-PSO-SCAC has better performance for convergence speed and search accuracy than the other two methods. From Fig. 10, it can be seen that H-PSO-SCAC has the best performance in this benchmark function, very similar to the f_8 function shown in Fig. 9. From Fig. 11, the H-PSO-SCAC algorithm shows lower search accuracy than the PSO method, but the H-PSO-SCAC algorithm has the fastest initial convergence speed.

Table 3 The solution accuracy (convergence iterations, means and standard deviations) of the function values

Function	Dim	k	PSO			H-PSO			H-PSO-SCAC		
			C.I.	Mean	S.D.	C.I.	Mean	S.D.	C.I.	Mean	S.D.
f_1	10	**1**	1000	6.2012×10^{-10}	6.6764×10^{-10}	339	4.8578×10^{-198}	0	648	**1.1602×10^{-254}**	**0**
	30	**1**	1000	3.7623×10^{-2}	1.3915×10^{-2}	643	4.7212×10^{-109}	2.1114×10^{-108}	686	**3.8701×10^{-120}**	**1.6838×10^{-119}**
	50	**1**	999	3.8913	7.3184×10^{-1}	697	4.1197×10^{-85}	1.8424×10^{-84}	969	**5.9074×10^{-99}**	**2.6419×10^{-98}**
f_2	10	**1**	1000	2.1333×10^{-6}	2.0538×10^{-6}	795	1.0746×10^{-112}	4.8058×10^{-112}	998	**6.3050×10^{-152}**	**2.8201×10^{-151}**
	30	0	1000	6.0732×10^{-2}	1.1693×10^{-2}	997	**7.7567×10^{-60}**	**3.4639×10^{-59}**	972	6.5128×10^{-55}	2.9126×10^{-54}
	50	**1**	1000	9.9744×10^{-1}	1.4371×10^{-1}	1000	3.1426×10^{-43}	1.4054×10^{-42}	861	**1.2217×10^{-54}**	**5.3981×10^{-54}**
f_3	10	0	1000	1.3354×10^{-5}	1.6915×10^{-5}	850	**9.0942×10^{-261}**	**0**	443	4.8132×10^{-209}	0
	30	**1**	1000	1.1100×10^{2}	4.4931×10^{1}	926	2.6831×10^{-113}	1.1999×10^{-112}	1000	**7.2841×10^{-117}**	**3.2575×10^{-116}**
	50	**1**	1000	3.2106×10^{3}	9.3806×10^{2}	798	1.0608×10^{-70}	4.7442×10^{-70}	656	**1.0190×10^{-71}**	**4.5572×10^{-71}**
f_4	10	**1**	1000	2.2019×10^{-3}	1.6108×10^{-3}	715	1.0093×10^{-145}	4.5138×10^{-145}	917	**6.5368×10^{-165}**	**0**
	30	**1**	1000	1.4001	2.7135×10^{-1}	830	5.2121×10^{-81}	2.3309×10^{-80}	316	**2.1974×10^{-105}**	**9.8269×10^{-105}**
	50	**1**	1000	5.6272	9.8911×10^{-1}	875	3.1305×10^{-68}	1.4000×10^{-67}	242	**0**	**0**
f_5	10	1	1000	**6.4367×10^{-2}**	**7.0667×10^{-2}**	1000	8.3245	2.5188×10^{-1}	993	8.4405	1.9672×10^{-1}
	30	**1**	1000	1.2644×10^{2}	1.0648×10^{2}	1000	2.8585×10^{1}	1.5425×10^{-1}	1000	**2.3598×10^{1}**	**1.5131×10^{-1}**
	50	**1**	1000	4.3369×10^{2}	3.1511×10^{2}	998	4.8693×10^{1}	2.3883×10^{-1}	999	**4.8608×10^{1}**	**1.5154×10^{-1}**
f_6	10	0	1000	1.9100×10^{-1}	4.6294×10^{-1}	8	**0**	**0**	6	**0**	**0**
	30	0	1000	8.5892×10^{1}	3.4675×10^{1}	9	**0**	**0**	10	**0**	**0**
	50	0	1000	4.5276×10^{2}	3.7128×10^{2}	10	**0**	**0**	9	**0**	**0**
f_7	10	0	1000	3.4692×10^{-24}	9.1195×10^{-24}	284	**0**	**0**	375	5.7180×10^{-298}	0
	30	**1**	1000	7.3742×10^{-10}	5.6203×10^{-10}	943	1.3760×10^{-184}	0	844	**3.9422×10^{-237}**	**0**
	50	**1**	1000	4.3871×10^{-6}	1.8612×10^{-6}	600	1.1154×10^{-132}	4.9880×10^{-132}	912	**8.8460×10^{-176}**	**0**
f_8	10	0	1000	9.0701×10^{-9}	1.7324×10^{-8}	21	**0**	**0**	23	**0**	**0**
	30	0	999	1.1057×10^{1}	2.7928	22	**0**	**0**	21	**0**	**0**
	50	0	1000	5.7297×10^{1}	1.0915×10^{1}	22	**0**	**0**	28	**0**	**0**
f_9	10	0	1000	1.2703×10^{-5}	9.4013×10^{-6}	33	**8.8818×10^{-16}**	**0**	36	**8.8818×10^{-16}**	**0**
	30	0	999	5.9541×10^{-2}	1.6829×10^{-2}	37	**8.8818×10^{-16}**	**0**	28	**8.8818×10^{-16}**	**0**
	50	0	1000	1.2146	3.5069×10^{-1}	54	**8.8818×10^{-16}**	**0**	71	**8.8818×10^{-16}**	**0**
f_{10}	10	0	1000	6.1674×10^{-2}	2.7533×10^{-2}	21	**0**	**0**	17	**0**	**0**
	30	0	1000	9.7200×10^{-2}	4.2895×10^{-2}	24	**0**	**0**	23	**0**	**0**
	50	0	1000	1.0200	3.8234×10^{-2}	34	**0**	**0**	21	**0**	**0**
f_{11}	10	0	1000	**1.4684×10^{-12}**	**2.4656×10^{-12}**	996	4.1352×10^{-2}	3.1200×10^{-2}	991	5.1324×10^{-2}	4.7964×10^{-2}
	30	0	1000	**1.2987×10^{-4}**	**8.0601×10^{-5}**	999	4.3793×10^{-1}	1.7885×10^{-1}	1000	4.1109×10^{-1}	1.3677×10^{-1}
	50	1	1000	**1.6937×10^{-1}**	**3.2479×10^{-1}**	998	7.4533×10^{-1}	2.6976×10^{-1}	998	9.0511×10^{-1}	2.2607×10^{-1}
f_{12}	10	0	1000	**3.3318×10^{-11}**	**5.9596×10^{-11}**	999	4.2521×10^{-1}	1.6747×10^{-1}	982	5.0929×10^{-1}	1.4419×10^{-1}
	30	1	999	**3.1048×10^{-3}**	**2.9633×10^{-3}**	992	2.3142	3.1010×10^{-1}	1000	2.3184	2.4370×10^{-1}
	50	1	1000	**5.6246×10^{-1}**	**1.6653×10^{-1}**	991	4.2424	2.3087×10^{-1}	999	4.4189	2.6983×10^{-1}

C.I.: convergence iteration; Mean: mean of objective values; S.D.: standard deviation.

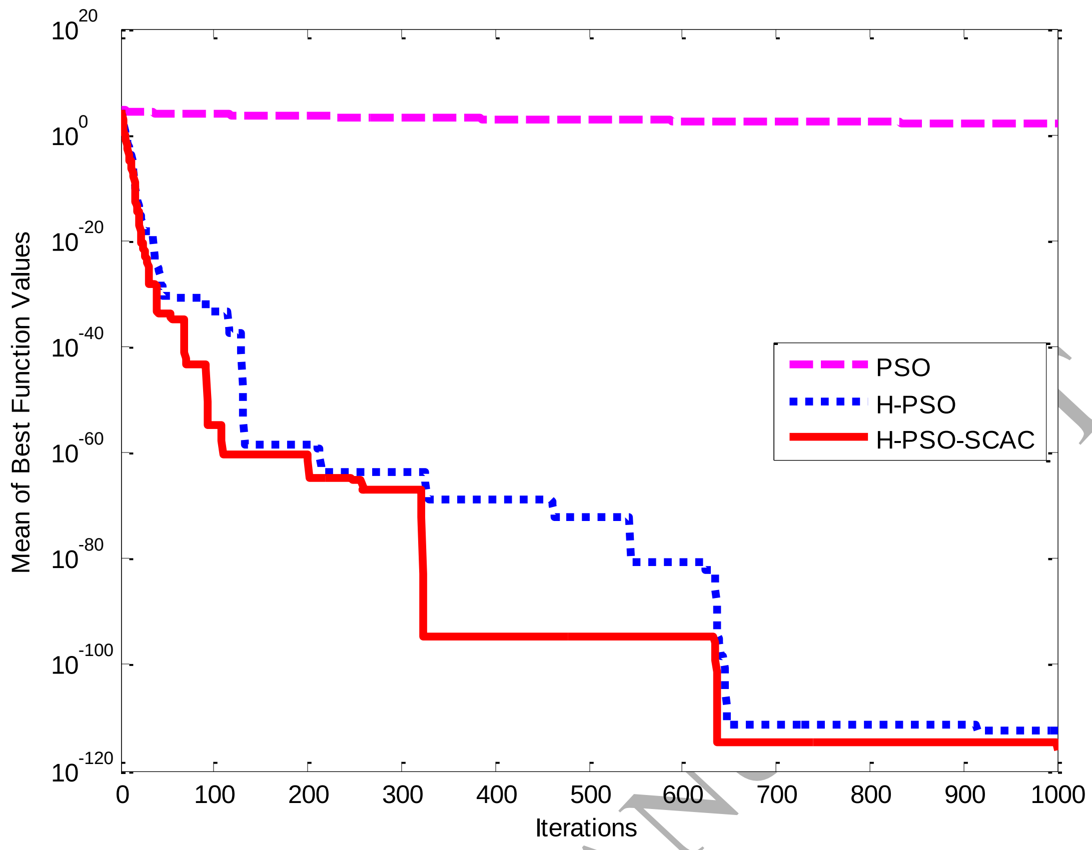

Fig. 6 Comparison of performances of three algorithms for minimization of f_3 with *Dim* = 30

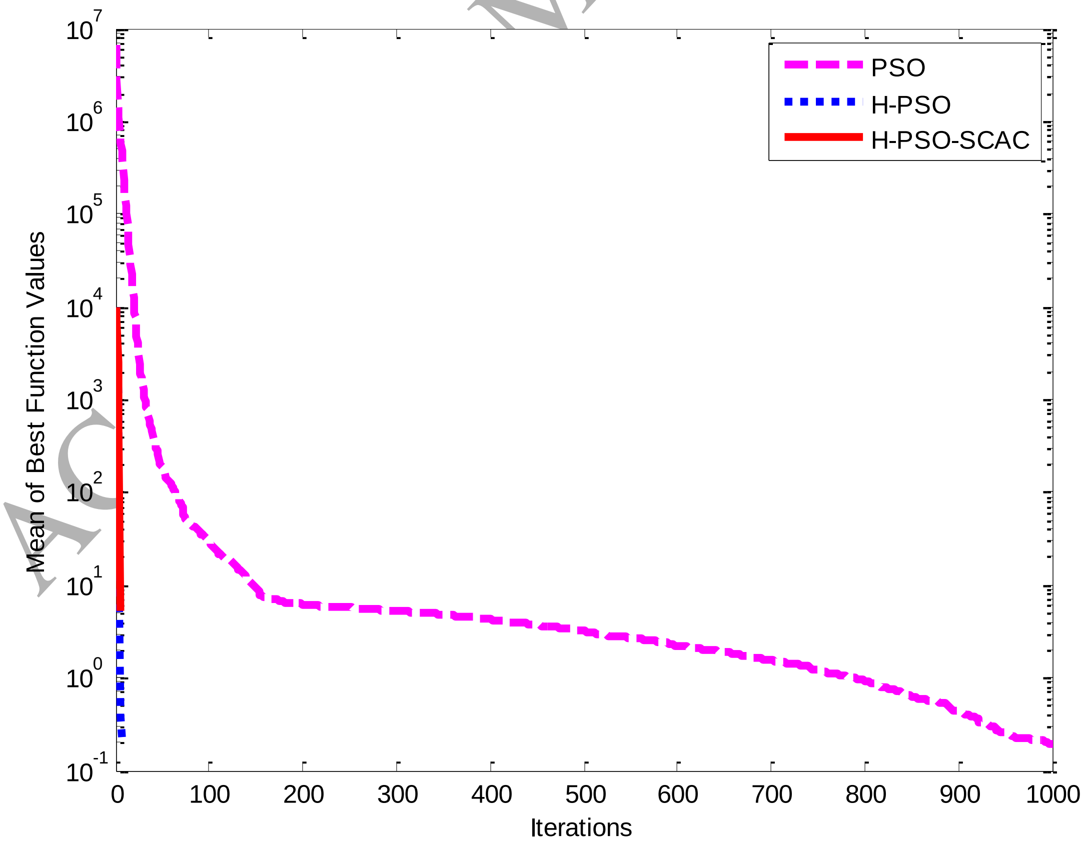

Fig. 7 Comparison of performances of three algorithms for minimization of f_6 with *Dim* = 10

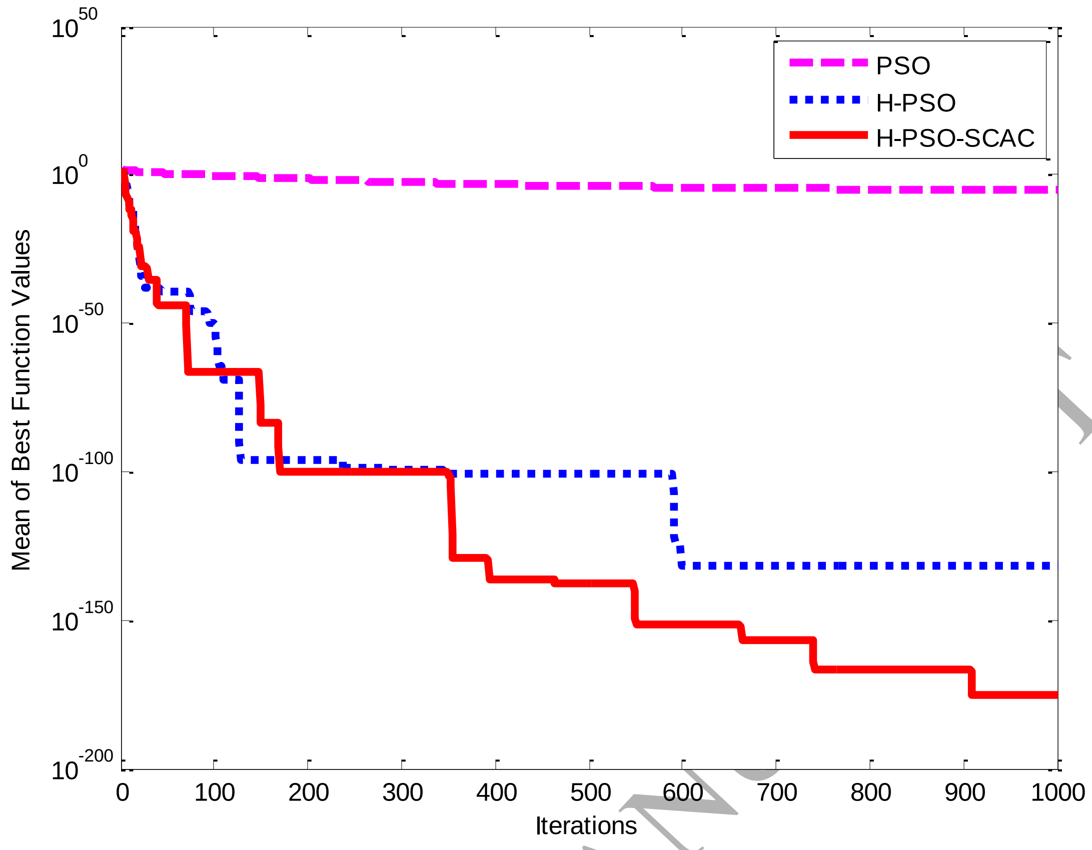

Fig. 8 Comparison of performances of three algorithms for minimization of f_7 with *Dim* = 50

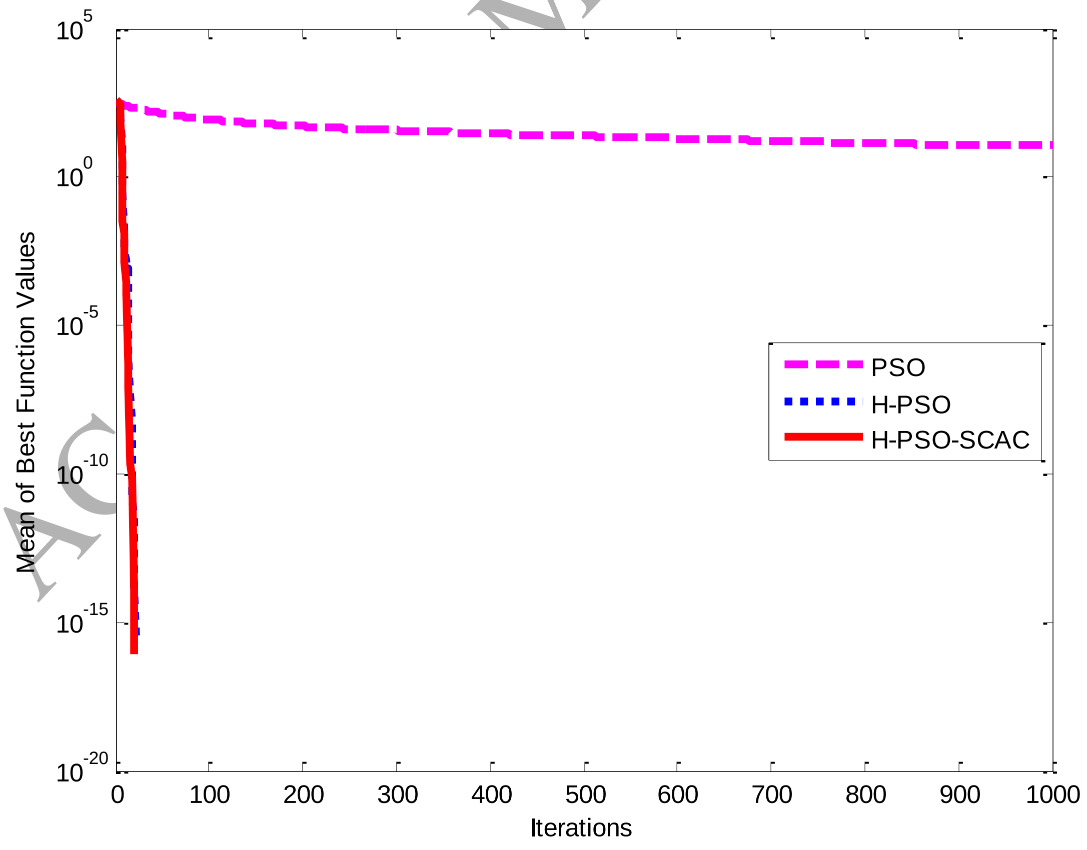

Fig. 9 Comparison of performances of three algorithms for minimization of f_8 with *Dim* = 30

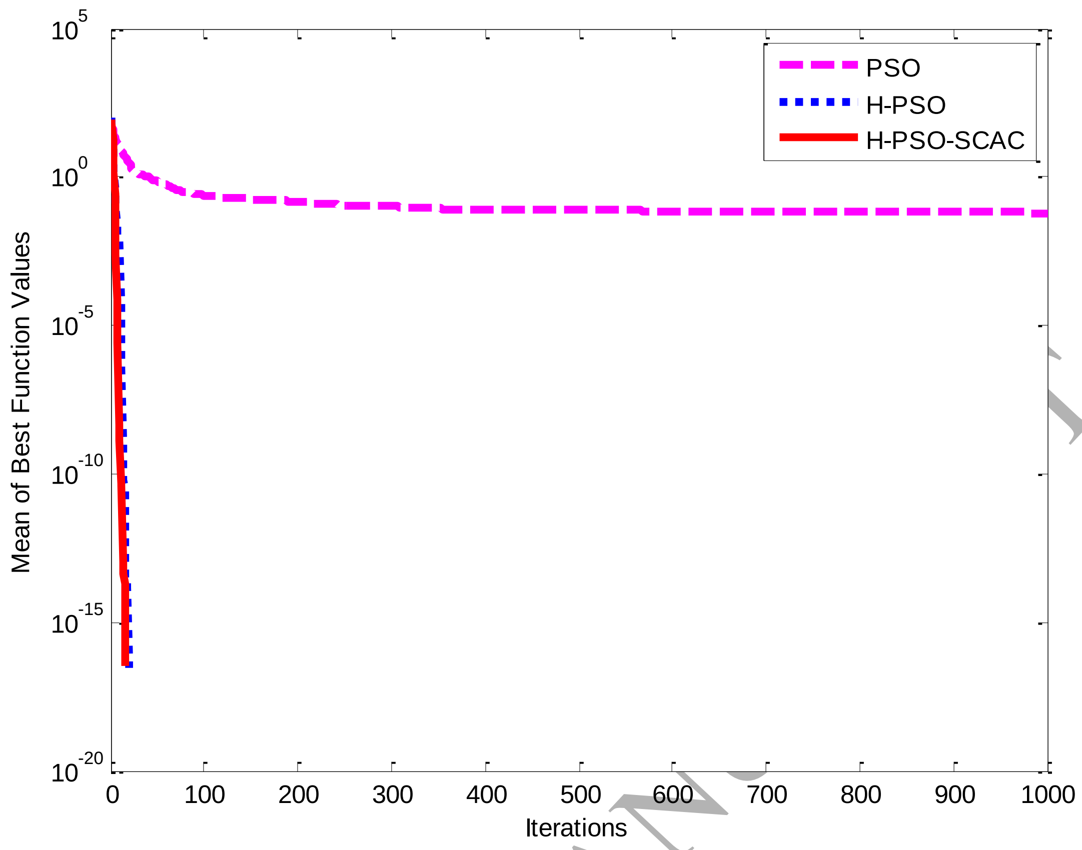

Fig. 10 Comparison of performances of three algorithms for minimization of f_{10} with *Dim* = 10

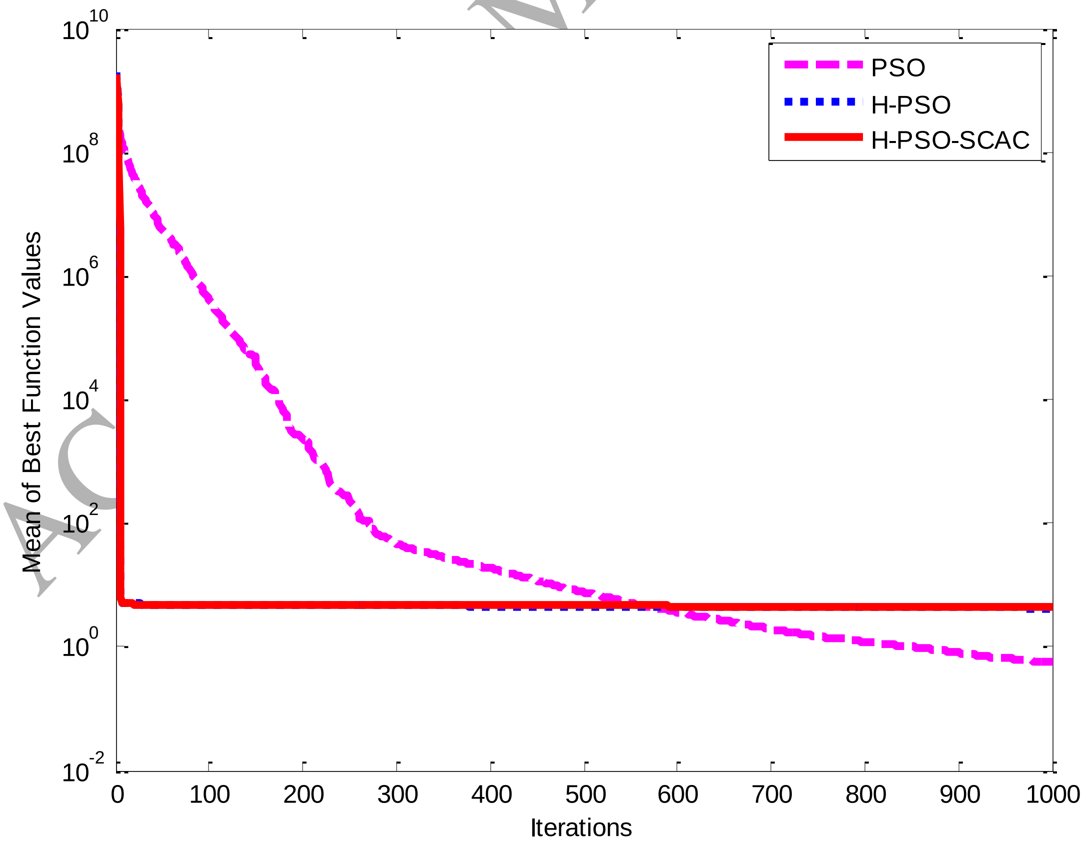

Fig. 11 Comparison of performances of three algorithms for minimization of f_{12} with *Dim* = 50

Through the above analysis, we can see that H-PSO-SCAC has better search performance and stronger ability to escape from a local optimum solution than PSO and H-PSO. Furthermore, based on the results of the *t*-tests, these results are different from the second best results. Therefore, the H-PSO-SCAC algorithm shows a very high efficiency on numerical optimization problems.

5.3 Results and comparison of H-PSO-SCAC and other similar algorithms

In this subsection, the H-PSO-SCAC algorithm is compared with the other optimization methods that have been proposed in recent years. Artificial bee colony (ABC), krill herd (KH) algorithm, biogeography-based optimization (BBO), sine cosine algorithm (SCA) and gravitational search algorithm (GSA) are compared with the proposed method. The parameter settings of the experiments for the six algorithms are shown in Table 4. Table 5 shows the mean results and the standard deviations of the ABC, KH, BBO, SCA, GSA and H-PSO-SCAC algorithms. Figs. 12 through 17 show the convergence curves of ABC, KH, BBO, SCA, GSA and H-PSO-SCAC algorithms, respectively.

Group A (unimodal functions): From Table 5, the H-PSO-SCAC algorithm performs better than the other methods on all unimodal functions (f_1, f_2, f_3, f_4, f_5, f_6 and f_7) when searching for global optimal solutions, except for function f_5 under Dim=10 and Dim=30. The ABC algorithm is the second most effective and is superior on function f_5 under Dim=10 and Dim=30. KH and BBO failed to acquire the best solution for any unimodal function. The SCA algorithm obtains better values for four functions (f_1, f_2, f_3 and f_4) under Dim=10. GSA yields better results for only three functions (f_1, f_2 and f_6), except for function f_2 under Dim=50.

Furthermore, the optimization processes of three unimodal functions (f_1, f_5 and f_7) of the six algorithms are given in Figs. 12 through 14, respectively. The values shown in these figures are the average numerical function optimum achieved from 20 executions. Fig. 12 presents the results obtained from six algorithms on the f_1 function, showing that the H-PSO-SCAC algorithm's performance is superior to the other five methods in the optimization process and shows better convergence accuracy and stronger ability to escape the local optimal. Fig. 13 shows the optimization results for the f_5 function. For this optimization problem, the figure shows that KH, SCA, GSA and BBO fail to determine a better solution. In addition, the ABC method shows higher convergence accuracy than the other five algorithms while H-PSO-SCAC shows faster convergence initially toward the global optimum. Fig. 14 shows the optimization values for the f_7 function, showing that H-PSO-SCAC apparently overtakes all other algorithms.

Group B (multimodal functions): Table 5 shows that, on average, the H-PSO-SCAC method performs better than the other five approaches on three multimodal functions (f_8, f_9 and f_{10}) when searching for the function minimum. ABC is the second most effective, performing better on function f_{11} under Dim=30 and Dim =50 and function f_{12} under Dim=50. GSA is the third most effective, performing better on function f_{11} under Dim=10 and function f_{12} under Dim=10 and Dim=30.

In addition, the convergence processes of three multimodal functions (f_8, f_9 and f_{11}) of the six approaches are given in Figs. 15 through 17, respectively. Fig. 15 shows the results obtained by the six algorithms on the f_8 function, which is a multimodal function and has many minor local optima. The H-PSO-SCAC approach could determine the theoretical global optima values. Fig. 16 illustrates the function values for the f_9 function, which is a difficult multimodal function. The figure shows that ABC, KH, BBO, SCA and GSA fail to locate better values in this multimodal function, while H-PSO-SCAC shows higher convergence accuracy and faster convergence speed than other approaches. Fig. 17 shows the optimization results for the f_{11} function. We observe that the H-PSO-SCAC algorithm fails to provide better results than the ABC and BBO approaches in this multimodal function, but H-PSO-SCAC shows faster convergence speed than the other five methods.

In summary, as seen from Table 5 and Figs. 12 through 17, the proposed H-PSO-SCAC algorithm shows stable search ability and better convergence accuracy than ABC, KH, BBO, SCA and GSA. Moreover, from the results in Table 5, it can be observed that the H-PSO-SCAC algorithm is significantly better than the other similar algorithms (except the f_{11} and f_{12} functions) for the 10, 30 and 50 dimension functions. This shows that the difference between the two methods is statistically significant. Therefore, it is concluded that the H-PSO-SCAC technique has much better search performance.

Table 4 Parameters settings of the six algorithms

Algorithm	Population	Maximum iterations	The dim of each object	other
ABC	40	1000	10,30,50	Limit=200
KH	40	1000	10,30,50	$N^{\max}=0.01, V_f=0.02, D^{\max}=0.005$
BBO	40	1000	10,30,50	Mu=0.005, $\mu=0.8$
SCA	40	1000	10,30,50	r_1, r_2, r_3 and r_4 are random numbers
GSA	40	1000	10,30,50	$G_0=100, \alpha=20$
H-PSO-SCAC	40	1000	10,30,50	$\omega=x_k=\frac{c}{4}\times\sin(\pi x_{k-1})$, $c_1=\partial\times\sin\left(\left(1-\frac{M_j}{M_{\max}}\right)\times\frac{\pi}{2}\right)+\delta$ $c_2=\partial\times\cos\left(\left(1-\frac{M_j}{M_{\max}}\right)\times\frac{\pi}{2}\right)+\delta$

Table 5 The means and standard deviations of the function values

Function	Dim	*k*	ABC		KH		BBO		SCA		GSA		H-PSO-SCAC	
			Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.
f_1	10	**1**	1.0074×10^{-16}	4.0454×10^{-17}	6.3350×10^{-4}	4.3871×10^{-4}	4.1840×10^{-2}	4.0736×10^{-2}	2.7766×10^{-28}	7.7425×10^{-28}	2.6425×10^{-18}	1.1316×10^{-18}	**1.1602×10^{-254}**	**0**
	30	**1**	4.4121×10^{-11}	3.3956×10^{-11}	1.7515×10^{-1}	8.0244×10^{-2}	1.5575	9.8810×10^{-1}	2.8447×10^{-3}	6.9847×10^{-3}	4.5361×10^{-17}	1.1024×10^{-17}	**3.8701×10^{-120}**	**1.6838×10^{-119}**
	50	**1**	5.6261×10^{-6}	1.0666×10^{-5}	8.8152×10^{-1}	2.6916×10^{-1}	1.2591×10^{1}	1.3988×10^{1}	6.8699×10^{1}	1.5411×10^{2}	1.5252×10^{-16}	3.4254×10^{-17}	**5.9074×10^{-99}**	**2.6419×10^{-98}**
f_2	10	**1**	3.8012×10^{-16}	8.9453×10^{-17}	1.0635×10^{-3}	1.2820×10^{-3}	2.6189×10^{-2}	2.0588×10^{-2}	4.6684×10^{-20}	1.4159×10^{-19}	4.5847×10^{-9}	9.2715×10^{-10}	**6.3050×10^{-152}**	**2.8201×10^{-151}**
	30	**1**	1.1187×10^{-6}	5.0686×10^{-7}	1.7345	1.2677	2.7942×10^{-1}	9.6707×10^{-2}	2.2332×10^{-5}	4.7211×10^{-5}	3.1787×10^{-8}	6.8259×10^{-9}	**6.5128×10^{-55}**	**2.9126×10^{-54}**
	50	**1**	1.5813×10^{-3}	5.6842×10^{-4}	7.2739	2.4338	9.6060×10^{-1}	1.5441×10^{-1}	1.3187×10^{-2}	3.0370×10^{-2}	2.1349×10^{-1}	6.7125×10^{-1}	**1.2217×10^{-54}**	**5.3981×10^{-54}**
f_3	10	**1**	5.3796×10^{1}	5.0998×10^{1}	8.3246×10^{-1}	6.0828×10^{-1}	1.4984×10^{2}	8.3144×10^{1}	6.7510×10^{-10}	2.9805×10^{-9}	6.2148×10^{-6}	2.7794×10^{-5}	**4.8132×10^{-209}**	**0**
	30	**1**	1.3553×10^{4}	2.0534×10^{3}	1.6941×10^{2}	5.6875×10^{1}	5.4877×10^{3}	1.8226×10^{3}	2.1156×10^{3}	1.7034×10^{3}	3.1879×10^{2}	1.1418×10^{2}	**7.2841×10^{-117}**	**3.2575×10^{-116}**
	50	**1**	4.3035×10^{4}	5.7221×10^{3}	9.1266×10^{2}	2.8919×10^{2}	1.9154×104	5.2972×10^{3}	3.0406×10^{4}	1.0254×10^{4}	1.3608×10^{3}	4.3799×10^{2}	**1.0190×10^{-71}**	**4.5572×10^{-71}**
f_4	10	**1**	1.9133×10^{-1}	1.2633×10^{-1}	1.8890×10^{-3}	3.3112×10^{-3}	6.8782×10^{-1}	1.9982×10^{-1}	3.9559×10^{-9}	1.1015×10^{-8}	1.1167×10^{-9}	2.1695×10^{-10}	**6.5368×10^{-165}**	**0**
	30	**1**	2.6374×10^{1}	4.2246	5.2557	2.1108	1.3774×10^{1}	4.0268	1.8041×10^{1}	1.1969×10^{1}	9.2726×10^{-2}	1.8637×10^{-1}	**2.1974×10^{-105}**	**9.8269×10^{-105}**
	50	**1**	5.4737×10^{1}	5.9766	8.8888	1.4942	2.8006×10^{1}	4.2939	6.0838×10^{1}	1.0391×10^{1}	5.6302	1.4417	**0**	**0**
f_5	10	1	**4.2952×10^{-1}**	**4.7747×10^{-1}**	1.1687×10^{1}	1.4494×10^{1}	1.1667×10^{2}	2.5166×10^{2}	7.1143	2.5489×10^{-1}	5.4964	1.4103×10^{-1}	8.4405	1.9672×10^{-1}
	30	1	**1.7817**	**1.7514**	8.3563×10^{1}	7.9227×10^{1}	4.0012×10^{2}	2.1139×10^{2}	1.6101×10^{2}	2.6356×10^{2}	26.1027	2.0043×10^{-1}	2.3598×10^{1}	1.5131×10^{-1}
	50	0	5.2165×10^{1}	3.6609×10^{1}	2.2123×10^{2}	1.7400×10^{2}	1.3330×10^{3}	6.8799×10^{2}	3.9556×10^{5}	7.7043×10^{5}	6.9783×10^{1}	4.5043×10^{1}	**4.8608×10^{1}**	**1.5154×10^{-1}**
f_6	10	**1**	1.1051×10^{-16}	5.2204×10^{-17}	5.0660×10^{4}	3.9406×10^{-4}	2.6853×10^{-2}	2.3240×10^{-2}	3.4748×10^{-1}	1.4054×10^{-1}	2.6455×10^{-18}	9.3311×10^{-19}	**0**	**0**
	30	**1**	2.9057×10^{-6}	2.3541×10^{-6}	1.5101×10^{-1}	7.5573×10^{-2}	2.4474	1.7506	4.4271	3.5361×10^{-1}	4.0717×10^{-17}	1.8505×10^{-17}	**0**	**0**
	50	**1**	7.7934×10^{-6}	1.3149×10^{-5}	7.2039×10^{-1}	2.5614×10^{-1}	7.6833	3.1716	6.5902×10^{1}	8.3698×10^{1}	1.5413×10^{-16}	4.6369×10^{-17}	**0**	**0**
f_7	10	**1**	2.0416×10^{-2}	6.8095×10^{-3}	5.4603×10^{-3}	3.4462×10^{-3}	2.0563×10^{-3}	3.9514×10^{-3}	1.2432×10^{-3}	1.2880×10^{-3}	6.2395×10^{-3}	2.9730×10^{-3}	**5.7180×10^{-298}**	**0**
	30	**1**	2.6315×10^{-1}	8.2781×10^{-2}	5.9401×10^{-2}	1.4211×10^{-2}	6.4414×10^{-2}	2.8352×10^{-2}	2.9719×10^{-2}	3.8158×10^{-2}	3.5543×10^{-2}	1.6153×10^{-2}	**3.9422×10^{-237}**	**0**
	50	**1**	7.4393×10^{-1}	1.2342×10^{-1}	1.7846×10^{-1}	3.7119×10^{-2}	3.4708×10^{-1}	1.2105×10^{-1}	5.1277×10^{-1}	6.2515×10^{-1}	2.4089×10^{-1}	1.8334×10^{-1}	**8.8460×10^{-176}**	**0**

f_8	10	0	**0**	**0**	4.2784	2.6834	1.0160×10^{2}	1.0671	2.1132×10^{-9}	9.4505×10^{-9}	4.6266	1.7762	**0**	**0**
	30	**1**	5.1774×10^{-1}	5.2577×10^{-1}	1.3903×10^{1}	9.2451	3.1610×10^{2}	2.8203	1.2694×10^{1}	2.0337×10^{1}	2.1939×10^{1}	6.8209	**0**	**0**
	50	**1**	7.4448	1.9741	1.9278×10^{1}	7.0329	5.3442×10^{2}	5.9494	7.2273×10^{1}	5.6043×10^{1}	4.9897×10^{1}	1.4556×10^{1}	**0**	**0**
f_9	10	**1**	9.4147×10^{-15}	2.4178×10^{-15}	76439×10^{-1}	1.1537	9.1512×10^{-2}	7.0473×10^{-2}	7.6383×10^{-15}	1.3536×10^{-14}	2.1734×10^{-9}	4.5479×10^{-10}	**8.8818×10^{-16}**	**0**
	30	**1**	1.7954×10^{-5}	9.9576×10^{-6}	5.0605	1.4602	5.5717×10^{-1}	3.8137×10^{-1}	1.0522×10^{1}	9.9199	5.0216×10^{-9}	6.3922×10^{-10}	**8.8818×10^{-16}**	**0**
	50	**1**	2.3159×10^{-2}	1.5955×10^{-2}	6.3114	1.4692	1.7222	3.3743×10^{-1}	1.8624×10^{1}	4.7575	7.7005×10^{-9}	1.1419×10^{-9}	**8.8818×10^{-16}**	**0**
f_{10}	10	**1**	5.4361×10^{-3}	7.6647×10^{-3}	1.0903×10^{-1}	5.4585×10^{-2}	1.0458×10^{-1}	4.2481×10^{-2}	6.2383×10^{-2}	1.5073×10^{-1}	3.6415×10^{-2}	6.2606×10^{-2}	**0**	**0**
	30	**1**	2.6519×10^{-3}	8.8454×10^{-3}	9.0618×10^{-2}	2.8149×10^{-2}	2.8142×10^{-1}	2.0290×10^{-1}	1.8881×10^{-1}	2.4642×10^{-1}	5.6433	2.2134	**0**	**0**
	50	**1**	7.8661×10^{-3}	1.1275×10^{-2}	1.9390×10^{-1}	4.7779×10^{-2}	1.0998	8.8204×10^{-2}	1.5218	1.2624	2.3163×10^{1}	5.1719	**0**	**0**
f_{11}	10	1	1.0166×10^{-16}	3.8798×10^{-17}	1.0249	1.1035	2.2001×10^{-3}	4.8006×10^{-3}	7.2370×10^{-2}	3.7025×10^{-2}	**5.3512×10^{-20}**	**2.0415×10^{-20}**	5.1324×10^{-2}	4.7964×10^{-2}
	30	1	**2.9534×10^{-12}**	**4.7247×10^{-12}**	3.7028	1.7669	2.7006×10^{-2}	3.3983×10^{-2}	9.4187×10^{-1}	7.3544×10^{-1}	3.5602×10^{-2}	7.3620×10^{-2}	4.1109×10^{-1}	1.3677×10^{-1}
	50	1	**4.2356×10^{-7}**	**4.7965×10^{-7}**	4.1634	8.9551×10^{-1}	1.3797×10^{-1}	1.5682×10^{-1}	1.3555×10^{6}	2.9909×10^{6}	9.6896×10^{-1}	5.1461×10^{-1}	9.0511×10^{-1}	2.2607×10^{-1}
f_{12}	10	1	1.8502×10^{-9}	3.3634×10^{-9}	7.0992×10^{-6}	8.5244×10^{-6}	6.6163×10^{-3}	6.9851×10^{-3}	2.3513×10^{-1}	6.3064×10^{-2}	**1.2148×10^{-32}**	**1.4556×10^{-32}**	5.0929×10^{-1}	1.4419×10^{-1}
	30	1	2.7256×10^{-8}	2.9729×10^{-8}	2.1867×10^{-4}	4.2770×10^{-4}	1.4260×10^{-1}	7.4042×10^{-2}	3.5488	1.9763	**9.1240×10^{-32}**	**1.4357×10^{-31}**	2.3184	2.4370×10^{-1}
	50	1	**5.2223×10^{-6}**	**5.5914×10^{-6}**	2.2187×10^{-1}	3.3949×10^{-1}	7.7534×10^{-1}	3.2384×10^{-1}	3.1732×10^{6}	5.6513×10^{6}	6.9656×10^{-2}	2.1491×10^{-1}	4.4189	2.6983×10^{-1}

Mean: mean of objective values; S.D.: standard deviation.

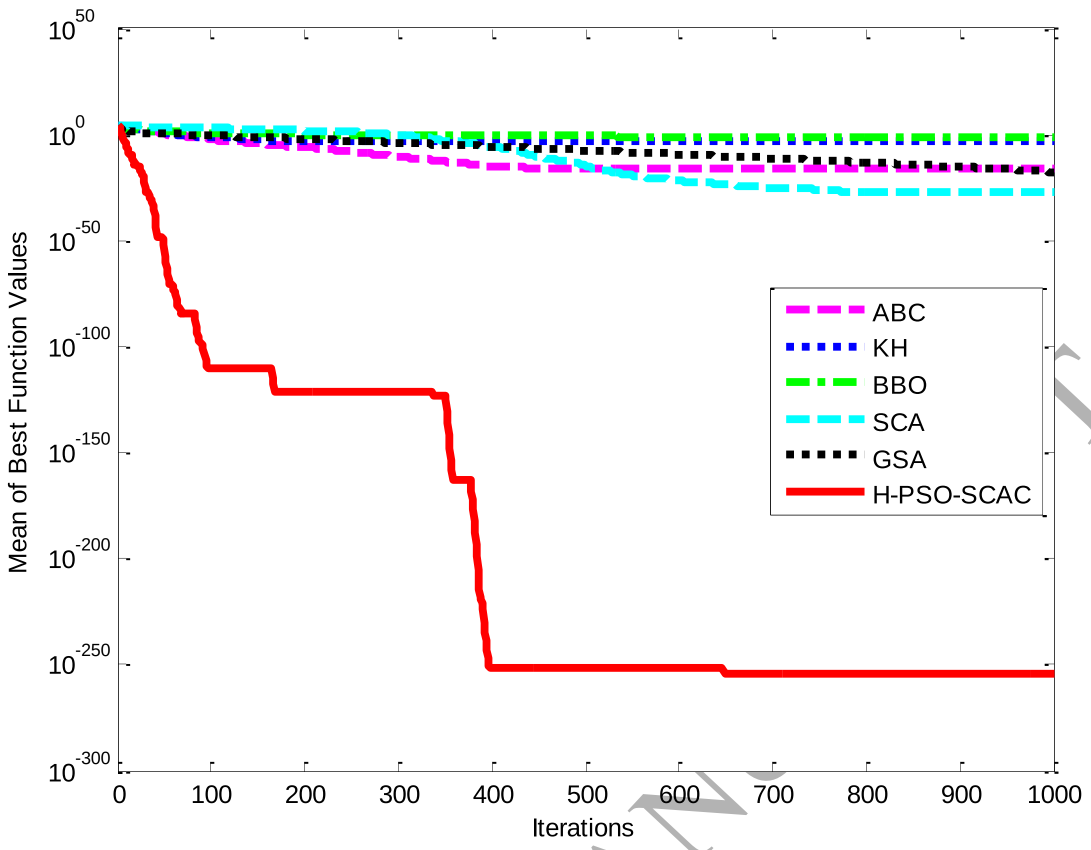

Fig. 12 Comparison of performances of six algorithms for minimization of f_1 with *Dim* = 10

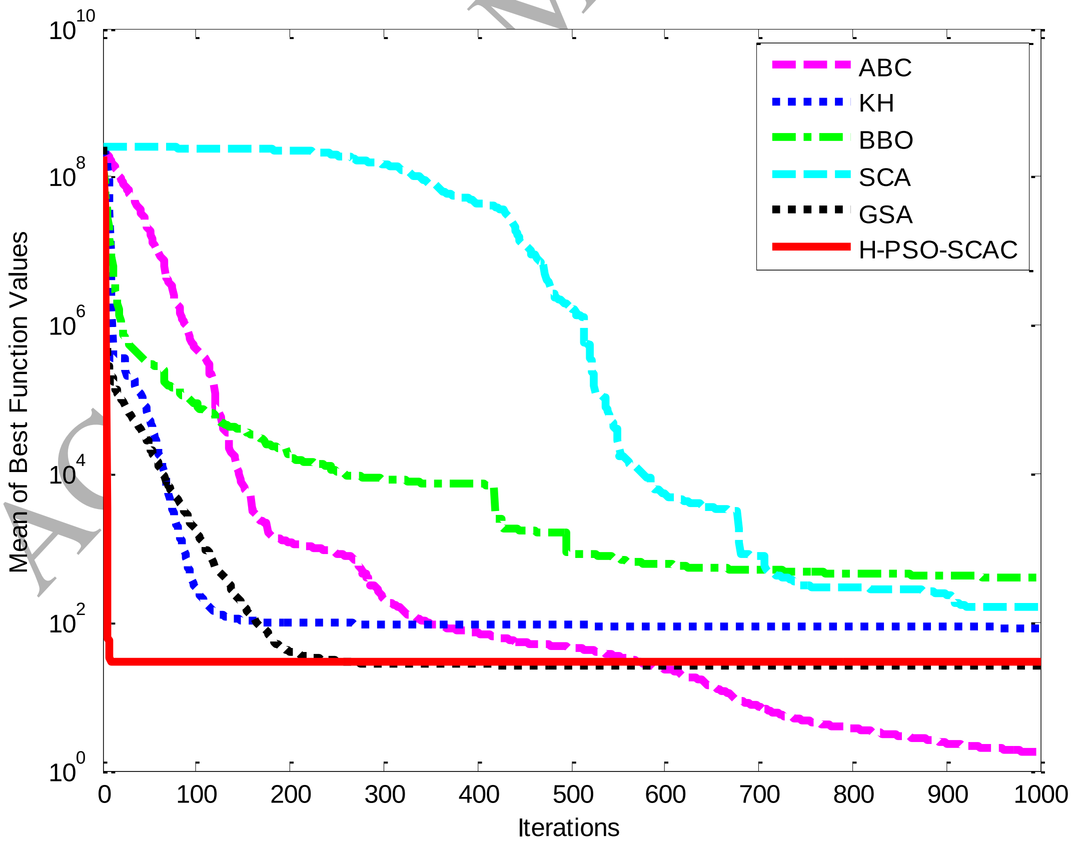

Fig. 13 Comparison of performances of six algorithms for minimization of f_5 with *Dim* = 30

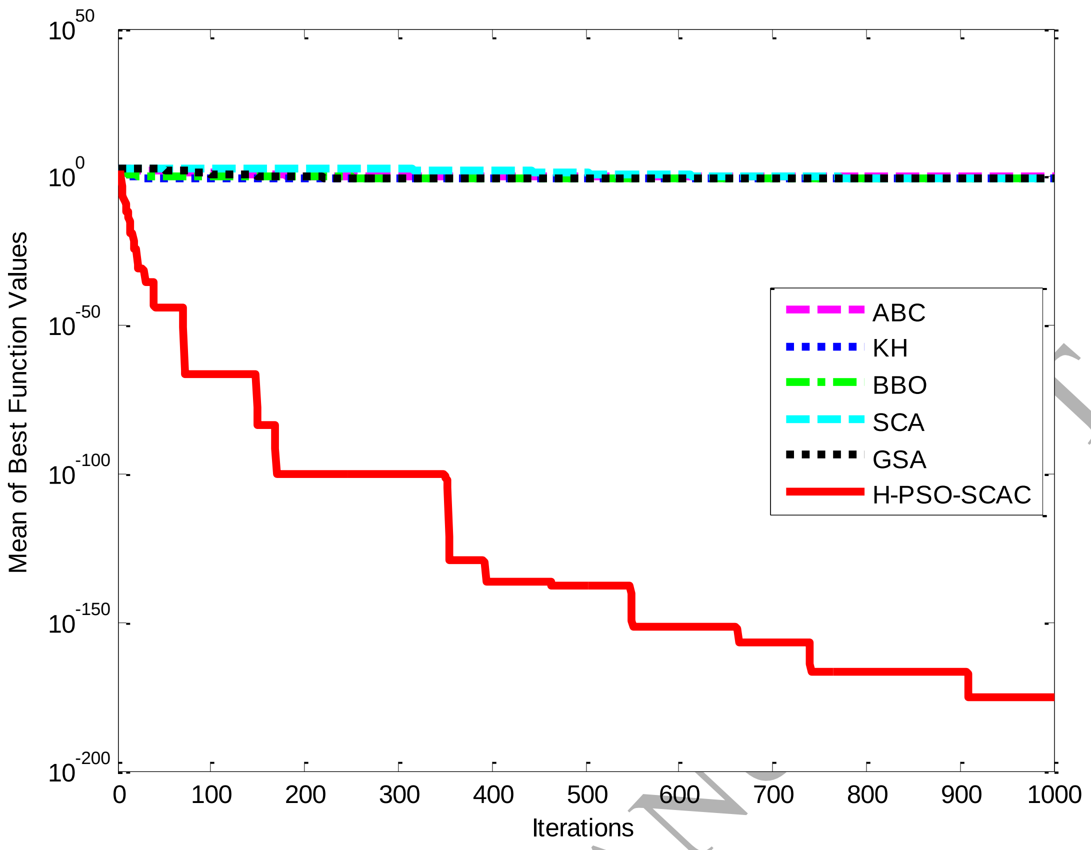

Fig. 14 Comparison of performances of six algorithms for minimization of f_7 with *Dim* = 50

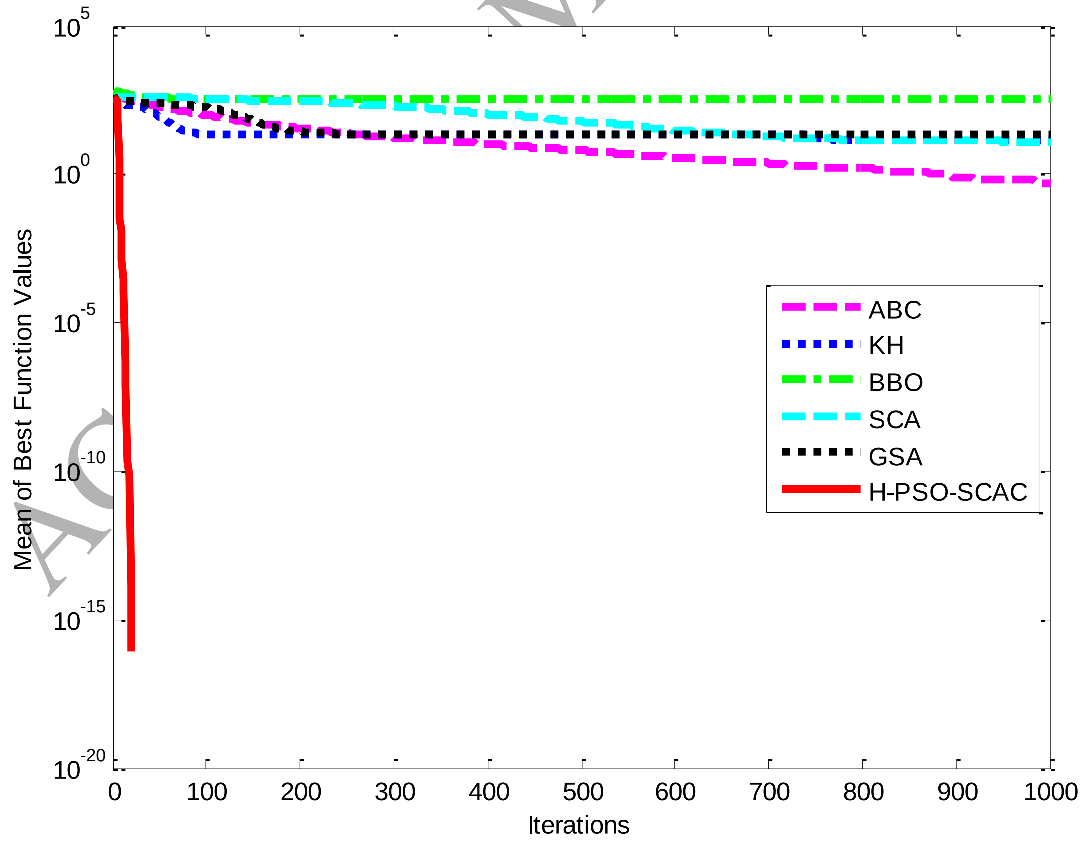

Fig. 15 Comparison of performances of six algorithms for minimization of f_8 with *Dim* = 30

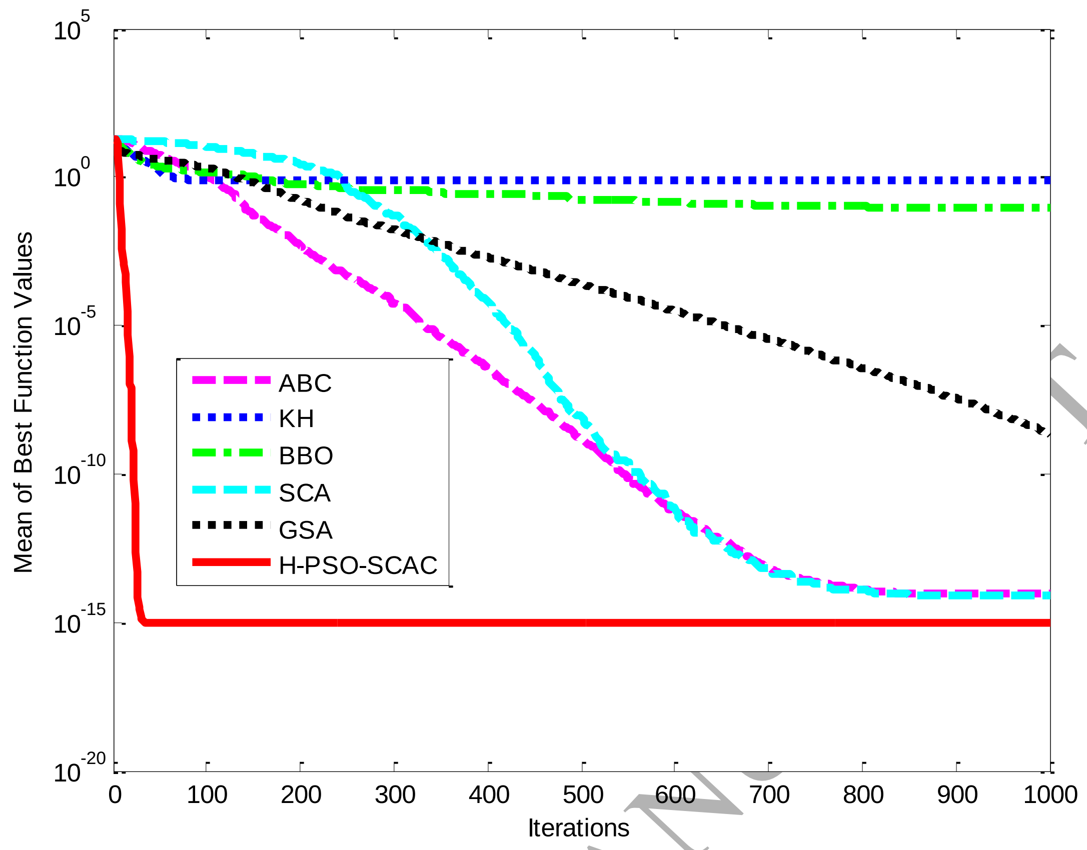

Fig. 16 Comparison of performances of six algorithms for minimization of f_9 with Dim = 10

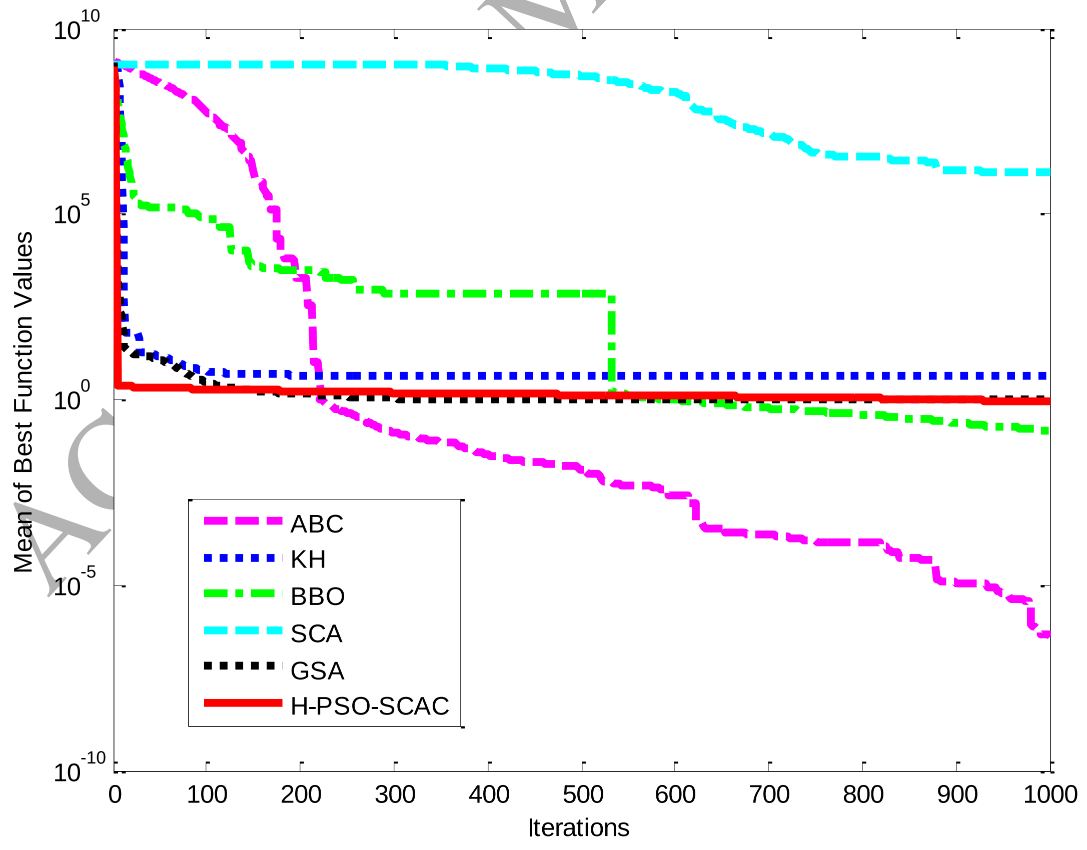

Fig. 17 Comparison of performances of six algorithms for minimization of f_{11} with Dim = 50

5.4 Results and comparisons of H-PSO-SCAC and PSO variants

In this subsection, the H-PSO-SCAC method is compared with other PSO variants, namely, GPSO, LPSO, SPSO-40, FIPSO, HPSO-TVAC, DMS-PSO, CLPSO, OPSO, OLPSO-G, ELPSO and LFPSO. The parameter settings in the experiments for the twelve algorithms are shown in Table 6. The results of the PSO variants algorithms in Table 7 are received from [49] and [16], while the results of ELPSO were obtained by experiments. The functions used to compare the approaches are four unimodal numerical functions (f_1, f_2, f_5, and f_7) and five multimodal numerical functions (f_8, f_9, f_{10}, f_{11} and f_{12}) described in subsection 4.1. The dimension of the nine test functions is 30. The best results among the twelve algorithms are shown in bold font.

From Table 7, it can be observed that the H-PSO-SCAC method achieves the best solution on the majority of the numerical functions. The CLPSO approach appears successful at unimodal numerical function f_5 and the OLPSO-G and LFPSO algorithms perform best on $f_{11}$and f_{12}, respectively. The remaining nine algorithms (GPSO, LPSO, SPSO-40, FIPS, HPSO-TVAC, DMS-PSO, OPSO, OLPSO-G and ELPSO) fail to determine better solutions than LFPSO, OLPSO-G and H-PSO-SCAC. Overall, H-PSO-SCAC performs best on f_1, f_2, f_7, f_8, f_9 and f_{10}. In the unimodal functions, H-PSO-SCAC is shown to offer superior performance among all the PSO variants, except CLPSO on the f_5 function. In the multimodal functions, H-PSO-SCAC generally outperforms all other PSO variants on three functions (f_8, f_9 and f_{10}).

Furthermore, Table 7 ranks the algorithms' performance in terms of their mean solution accuracy. It can be observed from the final ranking that H-PSO-SCAC offers the best overall performance, followed by LFPSO, OLPSO-G, HPSO-TVAC, GPSO, CLPSO, SPSO-40, ELPSO, OPSO, DMS-PSO, FIPS and LPSO.

Through analysis, the following conclusions can be drawn: Compared with 11 other algorithms, the H-PSO-SCAC approach can effectively increase the diversity of the search process and the chance of finding the global solution. Thus, H-PSO-SCAC can avoid premature convergence and being trapped in the local optimum. In summary, H-PSO-SCAC can be considered a very efficient optimization algorithm.

Table 6 PSO algorithms for comparison

Algorithms	Population size	Dim	Parameter settings	Reference
GPSO	40	30	ω : 0.9~0.4, $c_1 = c_2$ =2.0, $V_{\max}$ =0.2×Range	[42]
LPSO	40	30	ω :0.9~0.4, $c_1 = c_2$ =2.0, $V_{\max}$ =0.2×Range	[10]
SPSO-40	40	30	ω =0.721, $c_1 = c_2$ =1.193, K =3, without $V_{\max}$	[36]
FIPS	40	30	χ =0.729, $\sum c_i$ =4.1, $V_{\max}$ =0.5×Range	[32]
HPSO-TVAC	40	30	ω :0.9~0.4, c_1 :2.5~0.5, c_2 : 0.5~2.5, $V_{\max}$ =0.5×Range	[39]
DMS-PSO	40	30	ω : 0.9~0.2, $c_1 = c_2$ =2.0, m =3, R =5, $V_{\max}$ =0.2×Range	[50]
CLPSO	40	30	ω : 0.9~0.4, c =1.49445, m =7, $V_{\max}$ =0.2×Range	[27]
OPSO	40	30	ω : 0.9~0.4, $c_1 = c_2$ =2.0, $V_{\max}$ =0.5×Range	[17]
OLPSO-G	40	30	ω : 0.9~0.4, c =2.0, G =5, $V_{\max}$ =0.2×Range	[49]
LFPSO	40	30	ω : 1.0~0.0, $c_1 = c_2$ =2.0, $V_{\max}$ =0.2×Range	[16]
ELPSO	40	30	ω :1.0~0.0, $c_1 = c_2$ =2.0, $V_{\max}$ =0.2×Range	[22]
H-PSO-SCAC	40	30	ω : 1.0~0.0, c_1 :2.5~0.5, c_2 : 0.5~2.5, $V_{\max}$ =0.2×Range	--

Table 7 The means and standard deviations of the function values

f	Criteria	GPSO	LPSO	SPSO-40	FIPS	HPSO-TVAC	DMS-PSO	CLPSO	OPSO	OLPSO-G	LFPSO	ELPSO	H-PSO-SCAC
f_1	Mean	2.05×10^{-32}	3.34×10^{-14}	2.29×10^{-96}	2.42×10^{-13}	2.83×10^{-33}	2.65×10^{-31}	1.58×10^{-12}	6.45×10^{-18}	4.12×10^{-54}	4.69×10^{-31}	4.25×10^{-10}	**3.87×10^{-120}**
	S.D.	3.56×10^{-32}	5.39×10^{-14}	9.48×10^{-96}	1.73×10^{-13}	3.19×10^{-33}	6.25×10^{-31}	7.70×10^{-13}	4.64×10^{-18}	6.34×10^{-54}	2.50×10^{-30}	1.64×10^{-10}	**1.68×10^{-119}**
	Rank	5	9	2	10	4	6	11	8	3	7	12	**1**
f_2	Mean	1.49×10^{-21}	1.70×10^{-10}	1.74×10^{-53}	2.76×10^{-8}	9.03×10^{-20}	1.57×10^{-18}	2.51×10^{-8}	1.26×10^{-10}	9.85×10^{-30}	2.64×10^{-17}	3.59×10^{-9}	**6.51×10^{-55}**
	S.D.	3.60×10^{-21}	1.39×10^{-10}	1.58×10^{-53}	9.04×10^{-9}	9.58×10^{-20}	3.79×10^{-18}	5.84×10^{-9}	5.58×10^{-11}	1.01×10^{-29}	6.92×10^{-17}	1.78×10^{-9}	**2.91×10^{-54}**
	Rank	4	9	2	12	5	6	11	8	3	7	10	**1**
f_5	Mean	4.07×10^{1}	2.81×10^{1}	1.35×10^{1}	2.51×10^{1}	2.39×10^{1}	4.16×10^{1}	**3.82×10^{-4}**	4.96×10^{1}	2.15×10^{1}	2.38×10^{1}	7.23×10^{0}	2.36×10^{1}
	S.D.	3.22×10^{1}	2.18×10^{1}	1.46×10^{1}	5.10×10^{-1}	2.65×10^{1}	3.03×10^{1}	**1.28×10^{-7}**	3.65×10^{1}	2.99×10^{1}	3.17×10^{-1}	2.94×10^{0}	1.51×10^{-1}
	Rank	10	9	3	8	7	11	**1**	12	4	6	2	5
f_7	Mean	9.32×10^{-3}	2.28×10^{-2}	4.02×10^{-3}	4.24×10^{-3}	9.82×10^{-2}	1.45×10^{-2}	5.85×10^{-3}	5.50×10^{-2}	1.16×10^{-2}	2.41×10^{-3}	4.03×10^{-5}	**3.94×10^{-237}**
	S.D.	2.39×10^{-3}	5.60×10^{-3}	1.66×10^{-3}	1.28×10^{-3}	3.26×10^{-2}	5.05×10^{-3}	1.11×10^{-3}	1.70×10^{-3}	4.10×10^{-3}	8.07×10^{-4}	6.83×10^{-5}	**0**
	Rank	7	10	4	5	12	9	6	11	8	3	2	**1**
f_8	Mean	2.60×10^{1}	3.51×10^{1}	4.10×10^{1}	6.51×10^{1}	9.43×10^{0}	2.72×10^{1}	9.09×10^{-5}	6.97×10^{0}	1.07×10^{0}	4.54×10^{0}	5.63×10^{0}	**0**
	S.D.	7.27×10^{0}	6.89×10^{0}	1.11×10^{1}	1.34×10^{1}	3.48×10^{0}	6.02×10^{0}	1.25×10^{-4}	3.07×10^{0}	9.90×10^{-1}	1.03×10^{1}	4.87×10^{0}	**0**
	Rank	8	10	11	12	7	9	2	6	3	4	5	**1**
f_9	Mean	1.31×10^{-14}	8.20×10^{-8}	3.73×10^{-2}	2.33×10^{-7}	7.29×10^{-14}	1.84×10^{-14}	3.66×10^{-7}	6.23×10^{-9}	7.98×10^{-15}	1.68×10^{-14}	1.96×10^{-1}	**8.88×10^{-16}**
	S.D.	2.08×10^{-15}	6.73×10^{-8}	1.90×10^{-1}	7.19×10^{-8}	3.00×10^{-14}	4.35×10^{-15}	7.57×10^{-8}	1.87×10^{-9}	2.03×10^{-15}	4.84×10^{-15}	9.12×10^{-2}	**0**
	Rank	3	8	11	9	6	5	10	7	2	4	12	**1**
f_{10}	Mean	2.12×10^{-2}	1.53×10^{-3}	7.48×10^{-3}	9.01×10^{-12}	9.75×10^{-3}	6.21×10^{-3}	9.02×10^{-9}	2.29×10^{-3}	4.83×10^{-3}	8.14×10^{-17}	2.14×10^{-4}	**0**
	S.D.	2.18×10^{-2}	4.32×10^{-3}	1.25×10^{-2}	1.84×10^{-11}	8.33×10^{-3}	8.14×10^{-3}	8.57×10^{-9}	5.48×10^{-3}	8.63×10^{-3}	4.46×10^{-16}	1.23×10^{-4}	**0**
	Rank	12	6	10	3	11	9	4	7	8	2	5	**1**
f_{11}	Mean	2.23×10^{-31}	8.10×10^{-16}	7.47×10^{-2}	1.96×10^{-15}	2.71×10^{-29}	2.51×10^{-30}	6.45×10^{-14}	1.56×10^{-19}	**1.59×10^{-32}**	4.67×10^{-31}	7.17×10^{-3}	4.11×10^{-1}
	S.D.	7.07×10^{-31}	1.07×10^{-5}	3.11×10^{0}	1.11×10^{-11}	1.88×10^{-29}	1.02×10^{-29}	3.70×10^{-14}	1.67×10^{-19}	**1.03×10^{-33}**	9.01×10^{-31}	5.08×10^{-3}	1.37×10^{-1}
	Rank	2	7	11	8	5	4	9	6	**1**	3	10	12
f_{12}	Mean	1.32×10^{-3}	3.26×10^{-13}	1.76×10^{-3}	2.70×10^{-14}	2.79×10^{-28}	2.64×10^{-3}	1.25×10^{-12}	1.46×10^{-18}	4.39×10^{-4}	**1.51×10^{-28}**	4.78×10^{-4}	2.32
	S.D.	3.64×10^{-3}	3.70×10^{-3}	4.11×10^{-3}	1.57×10^{-14}	2.18×10^{-28}	4.79×10^{-3}	9.45×10^{-13}	1.33×10^{-18}	2.20×10^{-3}	**8.00×10^{-28}**	5.06×10^{-4}	2.44×10^{-1}
	Rank	9	5	10	4	2	11	6	3	7	**1**	8	12
Ave. rank		6.67	8.11	7.11	7.89	6.56	7.78	6.67	7.56	4.33	4.11	7.33	3.89
Final rank		5	12	7	11	4	10	5	9	3	2	8	1

Mean: mean of objective values; S.D.: standard deviation.

6. Conclusion and future work

In this paper, a novel optimization algorithm, named hybrid PSO with sine cosine acceleration coefficients (H-PSO-SCAC), has been presented to overcome the traditional PSO method’s drawbacks of premature convergence, trapping in local minima and being unable to balance exploration and exploitation. In H-PSO-SCAC, the sine cosine acceleration coefficients, opposition-based learning, sine map and a new position update formula are used to modify the search process. To validate H-PSO-SCAC, twelve numerical optimization problems (seven unimodal and five multimodal problems) with different levels of difficulty and features are used to test the proposed algorithms.

We established four group simulation experiments to evaluate the achievements of the proposed methods. First, the optimization results of PSO-SCAC are compared with PSO and PSO-TVAC. The simulations show that the novel parameter update strategy of the sine cosine acceleration coefficients is effective. Next, the simulation results of H-PSO-SCAC are compared with PSO and H-PSO. It can be seen clearly that H-PSO-SCAC shows better performance with quicker convergence speed and higher convergence accuracy for the majority of the functions. In addition, it is also observed that the SCAC strategy can effectively enhance the performance of H-PSO. Then, H-PSO-SCAC is compared with the other evolutionary algorithms. Finally, the H-PSO-SCAC algorithm is compared with other state-of-the-art PSO variants. The simulation results of the third and fourth experiments clearly demonstrate that the proposed method produces better average results in almost all numerical optimization problems and is more robust than the majority of other methods. Therefore, H-PSO-SCAC can be considered a novel and efficient optimization algorithm.

However, the performance in terms of the average optimum solution for the f_{11} and f_{12} functions was found to be significantly poor with the H-PSO-SCAC strategy. This indicates that we should further avoid the local optimum with non-increasing uncertainty. Future work will be focused on two directions: (i) the continued expansion of the study of the H-PSO-SCAC algorithm and (ii) the application of the proposed algorithm to real-world engineering problems.

Acknowledgments

Project supported by National Natural Science Foundation of China (Grant No. 61375084 and Grant No.61773242), Key Program of Natural Science Foundation of Shandong Province (ZR2015QZ08), Key Program of Scientific and Technological Innovation of Shandong Province (2017CXGC0926), Key Research and Development Program of Shandong Province (2017GGX30133). In addition, the authors thank the anonymous reviewers for providing valuable comments to improve this paper.

References

[1] E.K. Aydoğan, Y. Delice, U. Özcan, C. Gencer, Balancing stochastic U-lines using particle swarm optimization, J. Intell. Manuf. (2016) 1-15.

[2] M.A. Ahandani, Opposition-based learning in the shuffled bidirectional differential evolution algorithm, Swarm Evol. Comput. 26 (2016) 64-85.

[3] I. BoussaïD, J. Lepagnot, P. Siarry, A survey on optimization metaheuristics, Inf. Sci. 237 (2013) 82-117.

[4] J.S. Chou, A.D. Pham, Nature-inspired metaheuristic optimization in least squares support vector regression for obtaining bridge scour information, Inf. Sci. 399 (2017) 64-80.

[5] L.Z. Cui, G.H. Li, Z.X. Zhu, Q.Z. Lin, Z.K Wen, N. Lu, K.C. Wong, J.Y. Chen, A novel artificial bee

colony algorithm with an adaptive population size for numerical function optimization, Inf. Sci. 414 (2017) 53-67.

[6] M. Clerc, J. Kennedy, The particle swarm-explosion, stability, and convergence in a multidimensional complex space, IEEE Trans. Evol. Comput. 6 (1) (2002) 58-73.

[7] S.M. Chen, Z.C. Huang, Multiattribute decision making based on interval-valued intuitionistic fuzzy values and particle swarm optimization techniques, Inf. Sci. 397 (2017) 206-218.

[8] M. Dorigo, M. Birattari, T. Stutzle, Ant colony optimization, IEEE Comput. Intell. M. 1 (4) (2006) 28-39.

[9] Y. Delice, E.K. Aydoğan, U. Özcan, M. S. İlkay, A modified particle swarm optimization algorithm to mixed-model two-sided assembly line balancing, J. Intell. Manuf. 28 (1) (2017) 23-36.

[10] W.B. Du, W. Ying, G. Yan, The Impact of Population Structure on Particle Swarm Optimization: A Network Science Perspective, in: International Conference in Swarm Intelligence, Springer, 2016, pp: 341-349.

[11] T. Eftimov, P. Korošec, B.K. Seljak, A Novel Approach to Statistical Comparison of Meta-heuristic Stochastic Optimization Algorithms using Deep Statistics, Inf. Sci. 417 (2017) 186-215.

[12] A.P. Engelbrecht, Particle swarm optimization with crossover: a review and empirical analysis, Artif. Intell. Rev. 45 (2) (2016) 131-165.

[13] S. Fong, R. Wong, A.V. Vasilakos, Accelerated PSO swarm search feature selection for data stream mining big data, IEEE Trans. Serv. Comput. 9 (1) (2016) 33-45.

[14] A.M. Ghaedi, M. Ghaedi, A.R. Pouranfard, A. Ansari, Z. Avazzadeh, A. Vafaei, I. Tyagi, S. Agarwal, V.K. Gupta, Adsorption of Triamterene on multi-walled and single-walled carbon nanotubes: Artificial neural network modeling and genetic algorithm optimization, J. Mol. Liq. 216 (2016) 654-665.

[15] A.H. Gandomi, A.H. Alavi, Krill herd: a new bio-inspired optimization algorithm, Commun. Nonlinear Sci. 17 (12) (2012) 4831-4845.

[16] H. Haklı, H. Uğuz, A novel particle swarm optimization algorithm with Levy flight, Appl. Soft Comput. 23 (2014) 333-345.

[17] S.Y. Ho, H.S. Lin, W.H. Liauh, S.J. Ho, OPSO: Orthogonal particle swarm optimization and its application to task assignment problems, IEEE Trans. Syst. Man Cy. A 38 (2) (2008) 288-298.

[18] A.R. Jordehi, Particle swarm optimisation (PSO) for allocation of FACTS devices in electric transmission systems: A review, Renew. Sustain. Ener. 52 (2015) 1260-1267.

[19] A.R. Jordehi, Time varying acceleration coefficients particle swarm optimisation (TVACPSO): A new optimisation algorithm for estimating parameters of PV cells and modules, Energy Convers. Manage. 129 (2016) 262-274.

[20] A.R. Jordehi, A review on constraint handling strategies in particle swarm optimization, Neural Comput. Appl. 26 (6) (2015) 1265-1275.

[21] A.R. Jordehi, J. Jasni, Parameter selection in particle swarm optimisation: a survey, J. Exp Theor. Artif. In. 25 (4) (2013) 527-542.

[22] A.R. Jordehi, Enhanced leader PSO (ELPSO): a new PSO variant for solving global optimisation problems, Appl. Soft Comput. 26 (2015) 401-417.

[23] J. Kenndy, R.C. Eberhart, Particle swarm optimization, in: Proceedings of IEEE international conference on neural networks, IEEE, 1995, pp: 1942-1948.

[24] D. Karaboga, B. Gorkemli, C. Ozturk, A comprehensive survey: artificial bee colony (ABC) algorithm and applications, Artif. Intell. Rev. 42 (1) (2014) 21-57.

[25] M. Lozano, C. García-Martínez, F.J. Rodríguez, H.M. Trujillo, Optimizing network attacks by artificial

bee colony, Inf. Sci. 377 (2017) 30-50.

[26] Y.Y. Lin, J.Y. Chang, C.T. Lin, Identification and prediction of dynamic systems using an interactively recurrent self-evolving fuzzy neural network, IEEE Trans. Neur. Net. Lear. 24 (2) (2013) 310-321.

[27] 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. 10 (3) (2006) 281-295.

[28] H. Liu, Z. Cai, Y. Wang, Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization, Appl. Soft Comput. 10 (2) (2010) 629-640.

[29] X. Liang, W. Li, Y. Zhang, M.C. Zhou, An adaptive particle swarm optimization method based on clustering. Soft Comput. 19 (2) (2015) 431-448.

[30] S. Mirjalili, SCA: a sine cosine algorithm for solving optimization problems, Knowl.-Based Syst. 96 (2016) 120-133.

[31] S.H. Mousavi-Avval, S. Rafiee, M. Sharifi, S. Hosseinpour, B.Notarnicola, G. Tassielli, P.A. Renzulli, Application of multi-objective genetic algorithms for optimization of energy, economics and environmental life cycle assessment in oilseed production, J. Clean. Prod. 140 (2017) 804-815.

[32] R. Mendes, J. Kennedy, J. Neves, The fully informed particle swarm: simpler, maybe better, IEEE Trans. Evol. Comput. 8 (3) (2004) 204-210.

[33] P. Niu, K. Chen, Y. Ma, X Li, A. Liu, G. Li, Model turbine heat rate by fast learning network with tuning based on ameliorated krill herd algorithm, Knowl.-Based Syst. 118 (2017) 80-92.

[34] Y.V. Pehlivanoglu, A new particle swarm optimization method enhanced with a periodic mutation strategy and neural networks, IEEE Trans. Evol. Comput. 17 (3) (2013) 436-452.

[35] T. Peram, K. Veeramachaneni, C.K. Mohan, Fitness-distance-ratio based particle swarm optimization, in: Proceedings of the 2003 IEEE, 2003, pp: 174-181.

[36] Particle Swarm Central [Online], Available: http://www.particleswarm.info.

[37] B.Y. Qu, P.N. Suganthan, S. Das, A distance-based locally informed particle swarm model for multimodal optimization, IEEE Trans. Evol. Comput. 17 (3) (2013) 387-402.

[38] E. Rashedi, H. Nezamabadi-Pour, S. Saryazdi, GSA: a gravitational search algorithm, Inf. Sci. 179 (13) (2009) 2232-2248.

[39] 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.

[40] J. Shamir, J. Rosen, U. Mahlab, H.J, Caulfied, Optimization methods for pattern recognition, in: International Society for Optics and Photonics, 2017, pp: 1026202-1026202-23.

[41] D. Simon, Biogeography-based optimization, IEEE Trans. Evol. Comput. 12 (6) (2008) 702-713.

[42] Y. Shi, R. Eberhart, A modified particle swarm optimizer, in: The 1998 IEEE International Conference on. IEEE, 1998, pp: 69-73.

[43] F. Shabbir, P. Omenzetter, Particle swarm optimization with sequential niche technique for dynamic finite element model updating, Comput.-Aided Civ. Inf. 30 (5) (2015) 359-375.

[44] P.N. Suganthan, Particle swarm optimiser with neighbourhood operator, in: Proceedings of the 1999 Congress, IEEE, 1999, pp: 1958-1962.

[45] S.H. Strogatz. Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering, Westview, 2014.

[46] I.C. Trelea, The particle swarm optimization algorithm: convergence analysis and parameter selection, Inf. Process. Lett. 85 (6) (2003) 317-325.

[47] D.B Van, A.P. Engelbrecht, A cooperative approach to particle swarm optimization, IEEE Trans. Evol. Comput. 8 (3) (2004) 225-239.

[48] Z.Y. Wang, H.L Xing, T.R. Li, Y. Yang, R. Qu, Y. Pan, A modified ant colony optimization algorithm for network coding resource minimization, IEEE Trans. Evol. Comput. 20 (3) (2016) 325-342.

[49] Z.H. Zhan, J. Zhang, Y. Li, Y.H. Shi, Orthogonal learning particle swarm optimization, IEEE Trans. Evol. Comput. 15 (6) (2011) 832-847.

[50] S.Z. Zhao, P.N. Suganthan, Q.K. Pan, M.F. Tasgetiren, Dynamic multi-swarm particle swarm optimizer with harmony search, Expert Syst. Appl. 38 (4) (2011) 3735-3742.