Contents lists available at ScienceDirect

Information Sciences

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

HEPSO: High exploration particle swarm optimization

M.J. Mahmoodabadi [a,*], Z. Salahshoor Mottaghi [b], A. Bagheri [c]

[a] *Department of Mechanical Engineering, Sirjan University of Technology, Sirjan, Iran*
[b] *Department of Computer Engineering, Faculty of Engineering, University of Guilan, Rasht, Iran*
[c] *Department of Mechanical Engineering, Faculty of Engineering, University of Guilan, Rasht, Iran*

A R T I C L E I N F O

Article history:
Received 6 February 2012
Received in revised form 12 January 2014
Accepted 9 February 2014
Available online xxxx

Keywords:
Particle swarm optimization
Bee colony
Multi-crossover
HEPSO

A B S T R A C T

Particle swarm optimization (PSO) is a heuristic optimization technique which was inspired by flocking and swarming behavior of birds and insects. Same as other swarm intelligent methods, this algorithm also has its own disadvantages, such as premature convergence and rapid loss of diversity. In this paper, a new optimization method based on the combination of PSO and two novel operators is introduced in order to increase the exploration capability of the PSO algorithm (HEPSO). The first operator is inspired by the multi-crossover mechanism of the genetic algorithm, and the second operator uses the bee colony mechanism to update the position of the particles. Various values for probabilities are examined to find a trade-off for the PSO, multi-crossover formulation, and bee colony operator. The performance of the hybrid algorithm is tested using several well-known benchmark functions. The comparative study confirms that HEPSO is a promising global optimization algorithm and superior to the recent variants of PSO in terms of accuracy, speed, robustness, and efficiency.

1. Introduction

The antiquity of optimization comes back to the year 300 B.C., when Euclid tried to find the minimum distance between a point and a line. Nowadays, the importance of optimization in different branches of science, engineering, industry, commerce, etc. is clear to anyone. In general, optimization algorithms are divided into two major categories; deterministic and stochastic. Deterministic optimization techniques use the strategy of successive search on the derivative of the objective function. Hence, these techniques are suitable for continues, convex, and differentiable problems. However, the most of optimization problems are non-continues, non-convex, and non-differentiable. To optimize such problems, the stochastic optimization methods should be used. In recent decades, many researchers have returned their attention to stochastic methods such as evolutionary algorithms. Particle Swarm Optimization (PSO) which belongs to the class of the evolutionary algorithms is in the center of this attention.

PSO which is introduced by Kennedy and Eberhart is one of the modern heuristic algorithms and inspired by natural flocking and swarming behavior of birds and fish [12]. It was developed through simulation of simplified social systems and is robust to solve the optimization problems [2]. The PSO technique can generate a high-quality solution with short calculation time and has more stable convergence characteristics in comparison with other evolutionary methods [6,18,24,29]. In this algorithm, each particle has a memory that saves the personal best position and updates its position with

* Corresponding author. Tel.: +98 9306118541.
E-mail address: Mahmoodabadi@sirjantech.ac.ir (M.J. Mahmoodabadi).

http://dx.doi.org/10.1016/j.ins.2014.02.150
0020-0255/© 2014 Elsevier Inc. All rights reserved.

the global best position until arrives at the optimum solution. The PSO algorithm and its variants have been recently used to solve many problems such as engineering and statistical problems [3,9,23,28,30].

The original version of PSO suffers from trapping in local minima and premature convergence. In the recent years, several approaches, such as HPSO-TVAC [22], FIPS [19], GPSO [26], LPSO [13], VPSO [14], DMS-PSO [15], CLPSO [16], and APSO [31], have been proposed to modify the performance of PSO. In the self-organizing Hierarchical Particle Swarm Optimizer With Time-Varying Acceleration Coefficients (HPSO-TVAC), two new strategies based on the Time-Varying Acceleration Coefficients (TVAC) were proposed. The concept of mutation and self-organization hierarchical were introduced to the PSO with TVAC. The time-varying acceleration coefficients and inertia weight factor were used to control the global search and convergence to the global best solution [22]. In the Fully Informed Particle Swarm (FIPS), a particle was simply affected by the knowledge of all its neighbors. In the other words, all neighbors were used to update the velocity instead of using the global best position (gbest) and the particle best position (pbest). Fitness value and the neighborhood size were two important factors to estimate the influence of each particle over its neighbors [19]. In the Dynamic Multi-Swarm Particle Swarm Optimizer (DMS-PSO), the neighborhood topology was dynamic and randomly assigned. It means that the topological structure changes in a dynamic way to improve the performance of the original version of PSO [15]. Shi and Eberhart introduced a linearly decreasing inertia weight to control the search ability of the algorithm [26,27]. Kennedy and Mendes investigated various population topologies and proposed the Von Neumann topological structure. In comparison with other configurations, the Von Neumann has shown a good performance [13,14]. In the Comprehensive Learning Particle Swarm Optimizer (CLPSO), all other particles' historical best information was used to update the particle's velocity [16]. DMS-PSO [15] and CLPSO [16] are to be well performed on multimodal problems. In the Adaptive Particle Swarm Optimizer (APSO), by developing a systematic design of the parameter adaption, the inertia weight and acceleration coefficients were controlled, and an elitist learning strategy was applied. APSO could perform a global search over the entire search space with fast convergence speed [31]. General speaking, many researchers have tried to modify the original version of PSO in order to achieve a better accuracy and higher speed. Similarly, in this paper, the HEPSO algorithm is proposed to leap from local optima and also to modify the converging process. Numerical results show that the proposed operators are valid methods and help the PSO algorithm to find the global optimum, robustly.

The rest of this paper is organized as follows: Section 2 gives a brief review on the PSO algorithm. In Section 3, the proposed operators are introduced. Section 4 introduces the combination of PSO and new operators in details. Experimental results and comparison studies to verify the capability of proposed algorithm are shown in Section 5. Finally, Section 6 concludes the paper.

2. Particle swarm optimization

Particle swarm optimization is a population-based evolutionary algorithm and motivated by the simulation of social behavior instead of survival of the fittest [12]. PSO was initially used for balancing weights in the neural networks [5] and rapidly became a very popular global optimizer [7,8].

In the PSO method, each candidate solution is associated with a velocity [5]. The candidate solutions are called particles, and the position of each particle is changed according to its own experience and that of its neighbors (velocity). It is expected that the particles will move toward better solution areas. Mathematically, the particles are manipulated according to the following equations:

$$\vec{v}_i(t+1) = w\vec{v}_i(t) + C_1 r_1(\vec{x}_{pbest_i} - \vec{x}_i(t)) + C_2 r_2(\vec{x}_{gbest} - \vec{x}_i(t)) \tag{1}$$

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1) \tag{2}$$

where $\vec{x}(t)$ and $\vec{v}_i(t)$ denote the position and velocity of particle i at iteration (time step) t. $r_1, r_2 \in [0, 1]$ are random values. C_1 is the cognitive learning factor and represents the attraction of a particle toward its own success. C_2 is the social learning factor and represents the attraction of a particle toward the success of the entire swarm. w is the inertia weight which is employed to control the impact of the previous history of velocities on the current velocity of particle i. The personal best position of particle i is $\vec{x}_{pbest_i}$, and $\vec{x}_{gbest}$ is the position of the best particle of the entire swarm.

With a large value of C_1 and a small value of C_2, particles are allowed to move around their personal best position $\vec{x}_{pbest_i}$. With a small value of C_1 and a large value of C_2, particles converge to the best particle of the entire swarm $\vec{x}_{gbest}$. The previous researches showed that the best solutions were determined when C_1 is linearly decreased and C_2 is linearly increased over the iterations [22] as below:

$$C_1 = C_{1i} - (C_{1i} - C_{1f})\left(\frac{t}{max\ iteration}\right) \tag{3}$$

$$C_2 = C_{2i} - (C_{2i} - C_{2f})\left(\frac{t}{max\ iteration}\right) \tag{4}$$

C_{1i} and C_{2i} are the initial values of the learning factors C_1 and C_2, respectively. C_{1f} and C_{2f} are the final values of the learning factors C_1 and C_2, respectively. t is the current iteration number, and *max iteration* is the maximum number of allowable iterations.

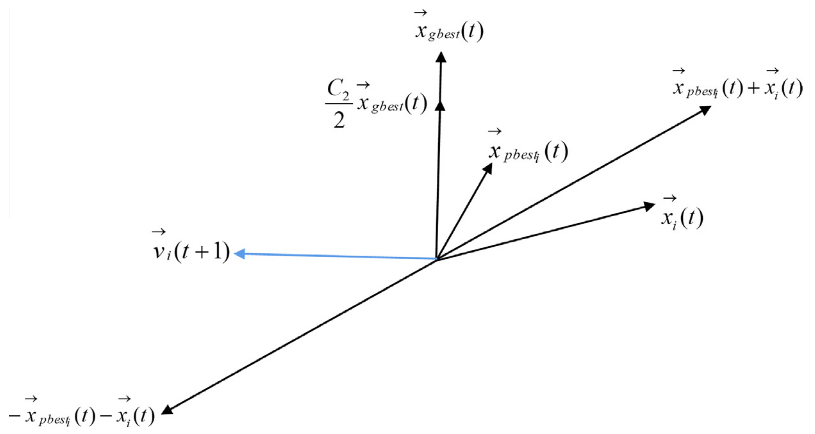
$\vec{x}_{gbest}(t)$
$\frac{C_2}{2}\vec{x}_{gbest}(t)$
$\vec{x}_{pbest_i}(t)+\vec{x}_i(t)$
$\vec{x}_{pbest_i}(t)$
$\vec{x}_i(t)$
$\vec{v}_i(t+1)$
$-\vec{x}_{pbest_i}(t)-\vec{x}_i(t)$

Fig. 1. A schematic diagram of the proposed multi-crossover model.

The inertia weight regulates the impact of the previous history of velocities on the current velocity of a particle [6]. A large inertia weight facilitates a global search while a small inertia weight makes possible a local search. By changing the inertia weight dynamically, the search ability is dynamically adjusted. Here, the adaptive inertia weight proposed in [31] is used which is given in Eq. (5).

$$W(f) = \frac{1}{1 + 1.5e^{-2.6f}} \in [0.4, 0.9] \tag{5}$$

where 0.4 and 0.9 are the initial and final values of the inertia weight, respectively. The evolutionary factor f is defined by Eq. (6).

$$f = \frac{d_g - d_{\min}}{d_{\max} - d_{\min}} \in [0, 1] \tag{6}$$

$$d_i = \frac{1}{N-1} \sum_{j=1, j\neq i}^{N} \sqrt{\sum_{k=1}^{D} (x_i^k - x_j^k)^2} \tag{7}$$

where d_i is the mean distance of particle i to all other particles that can be measured using an Euclidian metric via Eq. (7). N and D are the population size and the number of dimensions, respectively. d_i for the global best particle is shown as d_g. Also, by comparing all d_i 's, the maximum and minimum distances ($d_{\max}$ and $d_{\min}$) will be determine.

Numerical results show that the accelerating coefficients and inertia weight are major factors to achieve the good accuracy [22,31]. Nevertheless, combining PSO with other search techniques can also improve the algorithm ability [16]. For example, to increase the diversity of the population and to escape from local minimum, some evolutionary operators such as selection, crossover, and mutation have been combined with PSO [2,17,24]. Hence, in this paper, two evolutionary operators are used to improve the performance of the original version of PSO.

3. Proposed operators

In this section, two new formulae are introduced in which the first operator is based on multi-crossover [4], and the second performs similar to the bee colony mechanism [1]. These operators are used to improve the converging process and escape from local minima.

- **Operator 1.** The multi-crossover genetic algorithm was originally proposed by Chang as a novel method [4]. He used this algorithm for an especial case, to determine some parameters in a controller design problem. The multi-crossover genetic algorithm uses three parent chromosomes (Θ_1, Θ_2, Θ_3) unlike the classical crossover that uses only two chromosomes. If chromosome Θ_1 has the smallest fitness value, then it would be selected as the premier parent, and Eq. (8) would be used to generate new chromosome Θ_1'.

$$\Theta_1' = \Theta_1 + r(2\Theta_1 - \Theta_2 - \Theta_3) \tag{8}$$

where $r \in [0, 1]$ is a random value.

Here, we use this idea and propose a new operator that uses the best position of the group ($\vec{x}_{gbest}$) as the premier parent and the personal best position ($\vec{x}_{pbest_i}$) as the second parent. Hence, the following operator will generate the new velocity for the selected particle $\vec{x}_i(t)$ at random.

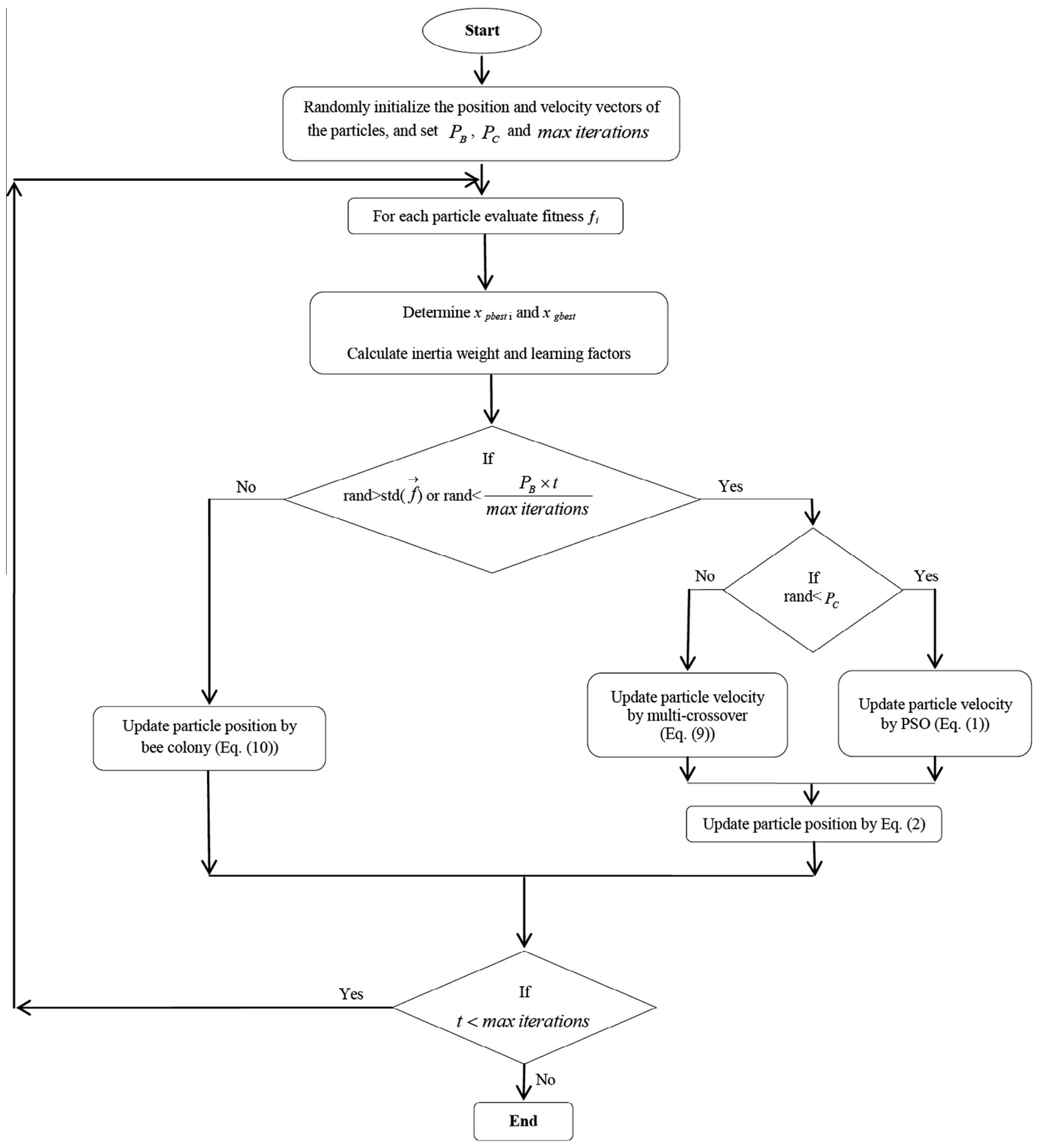
Start
Randomly initialize the position and velocity vectors of the particles, and set P_B, P_C and *max iterations*
For each particle evaluate fitness f_i
Determine $x_{pbest\,i}$ and x_{gbest}
Calculate inertia weight and learning factors
If rand>std($\vec{f}$) or rand< $\frac{P_B \times t}{max\ iterations}$
No
Yes
If rand< P_C
No
Yes
Update particle position by bee colony (Eq. (10))
Update particle velocity by multi-crossover (Eq. (9))
Update particle velocity by PSO (Eq. (1))
Update particle position by Eq. (2)
If $t <$ *max iterations*
Yes
No
End

Fig. 2. Flowchart of the HEPSO algorithm.

$$\vec{v}_i(t+1) = r\left(\frac{C_2}{2}\vec{x}_{gbest}(t) - \vec{x}_{pbest_i}(t) - \vec{x}_i(t)\right) \tag{9}$$

C_2 is the social learning factor (Eq. (4)), $\vec{x}_{pbest_i}$ is the personal best position, $\vec{x}_{gbest}$ is the global best position, and $r \in [0,1]$ is a random value. A schematic diagram of the proposed multi-crossover model is sketched in Fig. 1.

- **Operator 2.** The Artificial Bee Colony (ABC) algorithm proposed by Karaboga in 2005 is inspired by the forage behavior of the bee colony [11]. Employed, onlooker, and scout bees are simulated in the ABC algorithm. Each employed bee is associated with one food source site. The employed bee whose food source has been abandoned becomes a scout, and onlooker bees wait in the hive and decide on a food source to exploit based on the information shared by the employed bees [1,25]. The ABC algorithm has five steps; the first step; producing initial food source sites randomly. The second step; sending employed bees to the food source sites. An employed bee changes her position via local information and finds a

```
Initialize positions and velocities of particles in the swarm.
Set $P_B$ and $P_C$
For t= 1: maximum iterations
    For i=1: population size
        If $f(\vec{x}_i(t)) \le f(\vec{x}_{pbest_i}(t))$
            $x_{pbest_i}(t) = x_i(t)$
        Else
            $\vec{x}_{pbest_i}(t) = \vec{x}_{pbest_i}(t-1)$
        End
    End
    For i=1: population size
            $f(\vec{x}_{gbest}(t)) = \min f(\vec{x}_{pbest_i}(t))$
        If $(rand > standard\ deviation\ (f) \,||\, rand < \frac{P_B \times t}{max\ iterations})$
            If $rand < P_C$
                $\vec{v}_i(t+1) = w\,\vec{v}_i(t) + C_1 rand_1(\vec{x}_{pbest_i} - \vec{x}_i(t)) + C_2 rand_2(\vec{x}_{gbest} - \vec{x}_i(t))$
            Else
                $\vec{v}_i(t+1) = rand\,(\frac{C_2}{2}\vec{x}_{gbest}(t) - \vec{x}_{pbest_i}(t) - \vec{x}_i(t))$
            End
            $\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1)$
        Else
            $x_i^d(t+1) = x_i^d(t) + (2rand - 1)(x_i^d(t) - x_j^d(t))$
            $\vec{x}_{new} = \vec{x}_i(t)$
            $x_{new}^d = x_i^d(t+1)$
            If $f(\vec{x}_{new}) < f(\vec{x}_i(t))$
                $\vec{x}_i(t) = \vec{x}_{new}$
            End
        End
    End
End
```

Fig. 3. Pseudo code of the HEPSO algorithm.

neighboring food source. In fact, the fitness of the position is evaluated and if it has a better fitness, then it will be replaced with the old position. The third step; calculating probability values involved in probabilistic selection that it depends on the fitness values of the solutions in the population. Food source site selection by onlookers based on the information which is provided by employed bees is done in the fourth step; and in the final step, the abandonment criteria (limit and scout production) is checked [1].

Here, the food source finding operator of the ABC algorithm is used for the selected particles in the PSO strategy. New position $x_i(t+1)$ will be created by change in dth dimension of the randomly selected particle $x_i(t)$.

$$x_i^d(t+1) = x_i^d(t) + (2r - 1)(x_i^d(t) - x_j^d(t)) \tag{10}$$

$r \in [0, 1]$ is a random value. d is a integer random number in the range [1, dimension]. j is a integer random number in the range [1, number of particles]. After calculation of Eq. (10), the superior between $\vec{x}_i(t)$ and $\vec{x}_i(t+1)$ should be selected.

Table 1
Un-rotated unimodal and multimodal optimization test functions.

Name (comment)	Formula	Threshold	Search range
f_1: Sphere (unimodal)	$F(x) = \sum_{i=1}^{D} x_i^2$	0.01	$[-100, 100]^n$
f_2: Schwefel (unimodal)	$F(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	0.01	$[-10, 10]^n$
f_3: Quadric (unimodal)	$F(x) = \sum_{i=1}^{D} (\sum_{j=1}^{i} x_j)^2$	100	$[-100, 100]^n$
f_4: Rosenbrock (unimodal)	$F(x) = \sum_{i=1}^{D-1} \lfloor 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \rfloor$	100	$[-10, 10]^n$
f_5: Step (unimodal)	$F(x) = \sum_{i=1}^{D} (\lfloor x_i + 0.5 \rfloor)^2$	0	$[-100, 100]^n$
f_6: Quadric noise (unimodal)	$F(x) = \sum_{i=1}^{D} i x_i^4 + random[0, 1)$	0.01	$[-1.28, 1.28]^n$
f_7: Rastrigin (multimodal)	$F(x) = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10]$	50	$[-5.12, 5.12]^n$
f_8: Ackley (multimodal)	$F(x) = -20\exp\left(-0.2\sqrt{1/D\sum_{i=1}^{D} x_i^2}\right) - \exp\left(1/D\sum_{i=1}^{D} \cos(2\pi x_i)\right) + 20 + e$	0.01	$[-32, 32]^n$
f_9: Griewank (multimodal)	$F(x) = 1/4000\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos(x_i/\sqrt{i}) + 1$	0.01	$[-600, 600]^n$

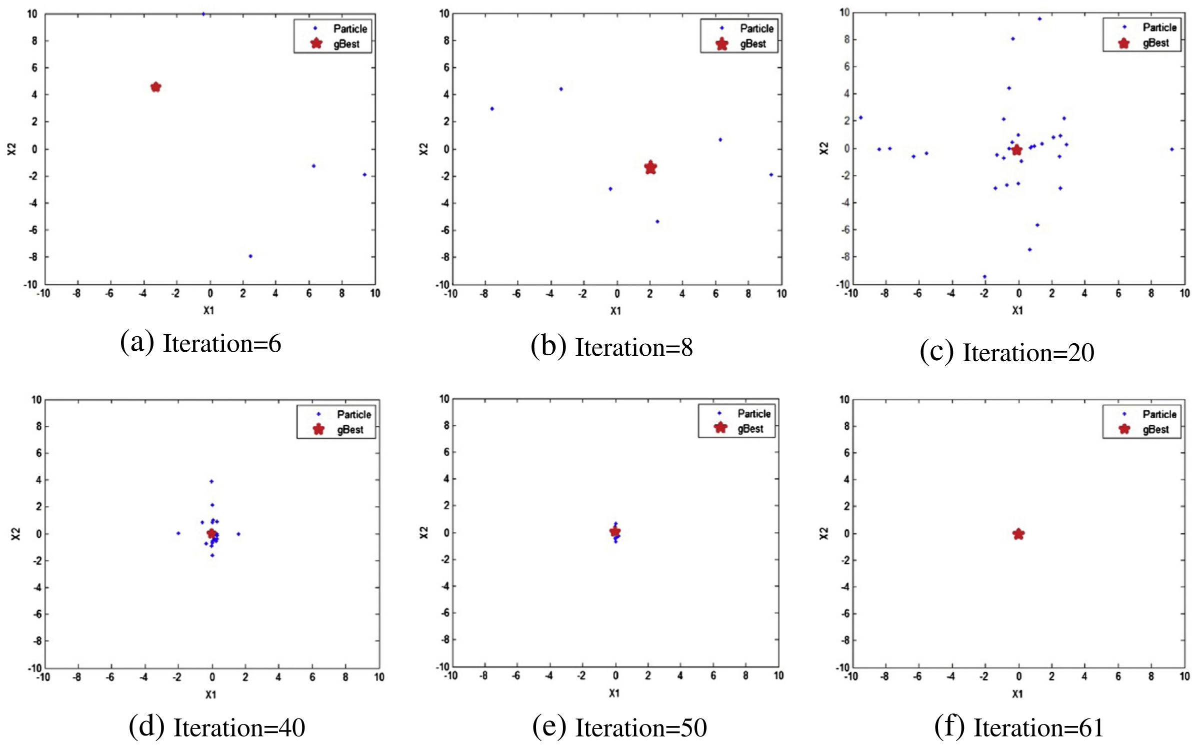

Fig. 4. The position of the particles observed in six stages of the HEPSO process for Sphere function, dimension = 2, and population = 50.

4. Hybrid of PSO and proposed operators

It is now possible to present a novel PSO (HEPSO) which is improved by utilizing two proposed operators to update the particle positions. Initially, the particles which form the population are randomly generated. In each iteration, the inertia weight (w) and the learning factors (C_1 and C_2) are calculated. Also, after calculation of the fitness values of all particles, $\vec{x}_{pbest_i}$ and $\vec{x}_{gbest}$ will be determined. Then, for each particle, two random numbers ρ_1, $\rho_1 \in [0, 1]$ would be allocated. If a particle does not have $\rho_1 > (standard\ deviation\ fitness\ values)$ or $\rho_2 < \frac{P_B \times t}{max\ iterations}$, then a new particle will be produced by the bee colony operator (Eq. (10)). P_B is the bee colony probability, t is the current iteration, and *max iterations* is the maximum number of allowable iterations. For each particle that is not chosen for the previous operation, another random number $\vartheta \in [0, 1]$ would be allocated. If a particle has $\vartheta < P_C$, then the multi-crossover operator would generate a velocity for new particle (Eq. (9)). P_C is the multi-crossover probability. Other particles that are not selected for the bee colony or multi-crossover operations will be enhanced by PSO (Eqs. (1) and (2)). This cycle should be repeated until the user-defined stopping criterion is satisfied. At the end, $\vec{x}_{gbest}$ would be determined as the best found solution. The flowchart and pseudo code of this

Table 2
Rotated unimodal and multimodal test functions.

Name (comment)	Formula	Optimum point	Search range
f_{10}: Shifted Sphere (Unimodal)	$F(x) = \sum_{i=1}^{D} z_i^2 + f\_bias$ $z = x - o, x = [x_1, x_2, \ldots, x_D], o = [o_1, o_2, \ldots, o_D]$	$x^* = o, F(x^*) = f\_bias = -450$	$x \in [-100, 100]^D$
f_{11}: Shifted Schwefel (Unimodal)	$F(x) = \sum_{i=1}^{D} (\sum_{j=1}^{i} z_j)^2 + f\_bias$ $z = x - o,\ x = [x_1, x_2, \ldots, x_D],\ o = [o_1, o_2, \ldots, o_D]$	$x^* = o, F(x^*) = f\_bias = -450$	$x \in [-100, 100]^D$
f_{12}: Shifted Rosenbrock (Unimodal)	$F(x) = \sum_{i=1}^{D-1} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + f\_bias$ $z = x - o,\ x = [x_1, x_2, \ldots, x_D],\ o = [o_1, o_2, \ldots, o_D]$	$x^* = o, F(x^*) = f\_bias = 390$	$x \in [-100, 100]^D$
f_{13}: Shifted Rastrigen (Multimodal)	$F(x) = \sum_{i=1}^{D} (z_i^2 - 10\cos(2\pi z_i) + 10) + f\_bias$ $z = x - o,\ x = [x_1, x_2, \ldots, x_D],\ o = [o_1, o_2, \ldots, o_D]$	$x^* = o, F(x^*) = f\_bias = -330$	$x \in [-5, 5]^D$
f_{14}: Shifted Rotated Ackley (Multimodal)	$F(x) = -20\exp\left(-0.2\sqrt{1/D\sum_{i=1}^{D} z_i^2}\right)$ $- \exp(1/D\sum_{i=1}^{D} \cos(2\pi z_i)) + 20 + e + f\_bias$ $z = (x - o) \times M,\ x = [x_1, x_2, \ldots, x_D],\ o = [o_1, o_2, \ldots, o_D]$	$x^* = o, F(x^*) = f\_bias = -140$	$x \in [-32, 32]^D$
f_{15}: Shifted Rotated Griewank (Multimodal)	$F(x) = \sum_{i=1}^{D} \frac{z_i^2}{4000} - -\prod_{i=1}^{D} \cos(\frac{z_i}{\sqrt{i}}) + 1 + f\_bias$ $z = (x - o) \times M,\ x = [x_1, x_2, \ldots, x_D],\ o = [o_1, o_2, \ldots, o_D]$	$x^* = o, F(x^*) = f\_bias = -180$	$x \in [0, 600]^D$

Table 3
Results comparison for original PSO, PSO with multi-crossover, PSO with bee colony, and HEPSO.

Function		Original PSO	PSO with bee colony	PSO with multi-crossover	HEPSO
f_1	Mean	$6.95 \times 10^{+3}$	1.63×10^{-6}	0	**0**
	Std. dev.	$2.71 \times 10^{+3}$	2.93×10^{-6}	0	**0**
f_2	Mean	3.2×10	5.51×10^{-2}	1.16×10^{-162}	$\mathbf{1.01 \times 10^{-162}}$
	Std. dev.	1.96	4.72×10^{-2}	0	**0**
f_4	Mean	$4.30 \times 10^{+4}$	$\mathbf{8.13 \times 10^{-2}}$	$1.97 \times 10^{+1}$	5.29
	Std. dev.	$3.22 \times 10^{+4}$	$\mathbf{7.24 \times 10^{-2}}$	2.28	6.01
f_5	Mean	$4.63 \times 10^{+3}$	0	0	**0**
	Std. dev.	$1.84 \times 10^{+3}$	0	0	**0**
f_6	Mean	2.12	9.84×10^{-2}	6.16×10^{-5}	$\mathbf{5.58 \times 10^{-5}}$
	Std. dev.	9.5×10^{-1}	3.15×10^{-2}	1.05×10^{-4}	$\mathbf{6.80 \times 10^{-5}}$
f_8	Mean	1.26×10	5.46×10^{-3}	1.60×10^{-15}	$\mathbf{1.60 \times 10^{-15}}$
	Std. dev.	1.23×10^{-2}	5.52×10^{-3}	1.59×10^{-15}	$\mathbf{1.59 \times 10^{-15}}$
f_9	Mean	5.55×10	8.02×10^{-2}	0	**0**
	Std. dev.	1.92×10	9.34×10^{-2}	0	**0**

Note: The bold values indicate the best results.

Table 4
The comparison of the probabilities to achieve the best performance of the HEPSO algorithm.

Bee colony probability (P_B)	**0.02**	0.04	0.05	0.06
Rosenbrock				
Mean	**5.29**	6.61	6.65	10.72
Std. dev.	**6.01**	9.79	6.65	12.91
Rastrrigen				
Mean	$\mathbf{2.92 \times 10^{-9}}$	1.41	1.40	1.59
Std. dev.	$\mathbf{6.52 \times 10^{-9}}$	2.62	1.35	3.56

Note: The bold values indicate the best results.

algorithm are shown in Figs. 2 and 3, respectively. The positions of the particles observed in six stages of the HEPSO process for the Sphere function (Table 1) with dimension 2 and population 50 are shown in Fig. 4. This figure shows that the population is distributed in the initial iterations. After 20 iterations, the particles gather round to the position of the best particle. At the end, in about 60 iterations, all particles can find the best solution.

5. Numerical results for the proposed algorithm

In this section, after introduce the un-rotated and rotated test functions, the new operators and the effects of its probabilities are analyzed. Then, the solution accuracy and the convergence speed of the algorithm are compared with those of the advanced evolutionary algorithms from the literature. For the proposed algorithm, C_1 is linearly decreased from $C_{1i} = 2.5$ to

Table 5
Results comparison on the algorithm accuracy for the un-rotated benchmark functions.

Function		GPSO	LPSO	VPSO	FIPS	HPSO-TVAC	DMS-PSO	CLPSO	APSO	HEPSO
f_1	Mean	1.98×10^{-53}	4.77×10^{-29}	5.11×10^{-38}	3.21×10^{-30}	3.38×10^{-41}	3.85×10^{-54}	1.89×10^{-19}	1.45×10^{-150}	**0**
	Std. dev.	7.08×10^{-53}	1.13×10^{-28}	1.91×10^{-37}	3.60×10^{-30}	8.50×10^{-41}	1.75×10^{-53}	1.49×10^{-19}	5.73×10^{-150}	**0**
f_2	Mean	2.51×10^{-34}	2.03×10^{-20}	6.29×10^{-27}	1.32×10^{-17}	6.9×10^{-23}	2.61×10^{-29}	1.01×10^{-13}	5.15×10^{-84}	$\mathbf{1.04 \times 10^{-162}}$
	Std. dev.	5.84×10^{-34}	2.89×10^{-20}	8.68×10^{-27}	7.86×10^{-18}	6.89×10^{-23}	6.6×10^{-29}	6.51×10^{-14}	1.44×10^{-83}	**0**
f_3	Mean	6.45×10^{-2}	18.60	1.44	0.77	2.89×10^{-7}	47.5	395	1.0×10^{-10}	$\mathbf{8.76 \times 10^{-52}}$
	Std. dev.	9.46×10^{-2}	30.71	1.55	0.86	2.97×10^{-7}	56.4	142	2.13×10^{-10}	$\mathbf{4.8 \times 10^{-51}}$
f_4	Mean	28.1	21.8627	37.6469	22.5387	13	32.3	11	2.84	**0.0736**
	Std. dev.	24.6	11.1593	24.9378	0.310182	16.5	24.1	14.5	3.27	**0.084**
f_5	Mean	0	0	0	0	0	0	0	0	**0**
	Std. dev.	0	0	0	0	0	0	0	0	**0**
f_6	Mean	7.77×10^{-3}	1.49×10^{-2}	1.08×10^{-2}	2.55×10^{-3}	5.54×10^{-2}	1.1×10^{-2}	3.92×10^{-3}	4.66×10^{-3}	$\mathbf{3.51 \times 10^{-5}}$
	Std. dev.	2.42×10^{-3}	5.66×10^{-3}	3.24×10^{-3}	6.25×10^{-4}	2.08×10^{-2}	3.94×10^{-3}	1.14×10^{-3}	1.7×10^{-3}	$\mathbf{4.84 \times 10^{-5}}$
f_7	Mean	30.7	34.90	34.09	29.98	2.39	28.1	2.57×10^{-11}	5.8×10^{-15}	**0**
	Std. dev.	8.64	7.25	8.07	10.92	3.71	6.42	6.64×10^{-11}	1.01×10^{-14}	**0**
f_8	Mean	1.15×10^{-14}	1.85×10^{-14}	1.4×10^{-14}	7.68×10^{-15}	2.06×10^{-10}	8.52×10^{-15}	2.01×10^{-12}	1.11×10^{-14}	$\mathbf{1.13 \times 10^{-15}}$
	Std. dev.	2.27×10^{-15}	4.80×10^{-15}	3.48×10^{-15}	9.33×10^{-16}	9.45×10^{-10}	1.79×10^{-15}	9.22×10^{-13}	3.55×10^{-15}	$\mathbf{9.01 \times 10^{-16}}$
f_9	Mean	2.37×10^{-2}	1.10×10^{-2}	1.31×10^{-2}	9.04×10^{-4}	1.07×10^{-2}	1.31×10^{-2}	6.45×10^{-13}	1.67×10^{-2}	**0**
	Std. dev.	2.57×10^{-2}	1.60×10^{-2}	1.35×10^{-2}	2.78×10^{-3}	1.14×10^{-2}	1.73×10^{-2}	2.07×10^{-12}	2.41×10^{-2}	**0**

Note: The bold values indicate the best results.

$C_{1f} = 0.5$, while C_1 is linearly increased from $C_{2i} = 0.5$ to $C_{2f} = 2.5$ over iteration. It can be noted that the calculations are performed in the environment of MATLAB software with accuracy of 10^{-330}.

5.1. Test functions

In Table 1, nine un-rotated benchmark functions are introduced. These benchmark functions are divided into two groups; unimodal and multimodal. Unimodal functions have only one global optimum without any local optimum but multimodal functions have a global minimum with numerous local minima.

The best solution for the test functions of Table 1 $f_{\min} = 0$ when $x_i = 0(i = 1, 2, \ldots, n)$, except to Rosenbrock function. In this test function, if $x_i = 1(i = 1, 2, \ldots, n)$, then the best solution $f_{\min} = 0$. Furthermore, in order to provide a more accurate assessment of the performance of HEPSO, six rotated benchmark functions are implemented. The definitions of these test functions are summarized in Table 2. The test functions should be minimized, and a strong algorithm can escape from local minima, and its particles are able to move to the global optimum or to the nearest position of the global optimum.

5.2. Examination of the new operators

In this section, four algorithms (Original PSO, PSO with multi-crossover, PSO with bee colony, and HEPSO) are compared to illustrate the individuals and combined effects of the proposed operators. The population size, maximum iteration, and dimension are set at 20, 5500, and 30, respectively. In PSO with multi-crossover, the bee colony operator is neglected and the multi-crossover probability is set at $P_C = 0.95$. Similarly, in PSO with bee colony, the multi-crossover operator is neglected and the bee colony probability is set at $P_B = 0.02$. The results are summarized in Table 3 (the bold values indicate the best results), and it can be understand from this table that the incorporation of these operators can greatly improve the performance of the original version of PSO.

5.3. Analysis of the effects of the probabilities

In this section, to find a trade-off for the percent usage of the PSO, multi-crossover formula, and bee colony operator, several possibilities have been examined. Furthermore, the PSO and multi-crossover formula have 95 and 2 percent probabilities, respectively. The mean and standard deviation fitness of the best particle for thirty independent runs are shown in Table 4 (the bold values indicate the best results). The experiences illustrated that the best solutions would be found, if the bee colony operator is used in the initial iterations.

5.4. Analysis of the solution accuracy of the algorithm

This section starts with the results comparison on the test functions of Table 1. Then, the convergence trajectories of the algorithms are compared. Furthermore, a comparative study between HEPSO and various evolutionary algorithms that are representative of the state-of-the-arts is conducted on the rotated benchmark functions.

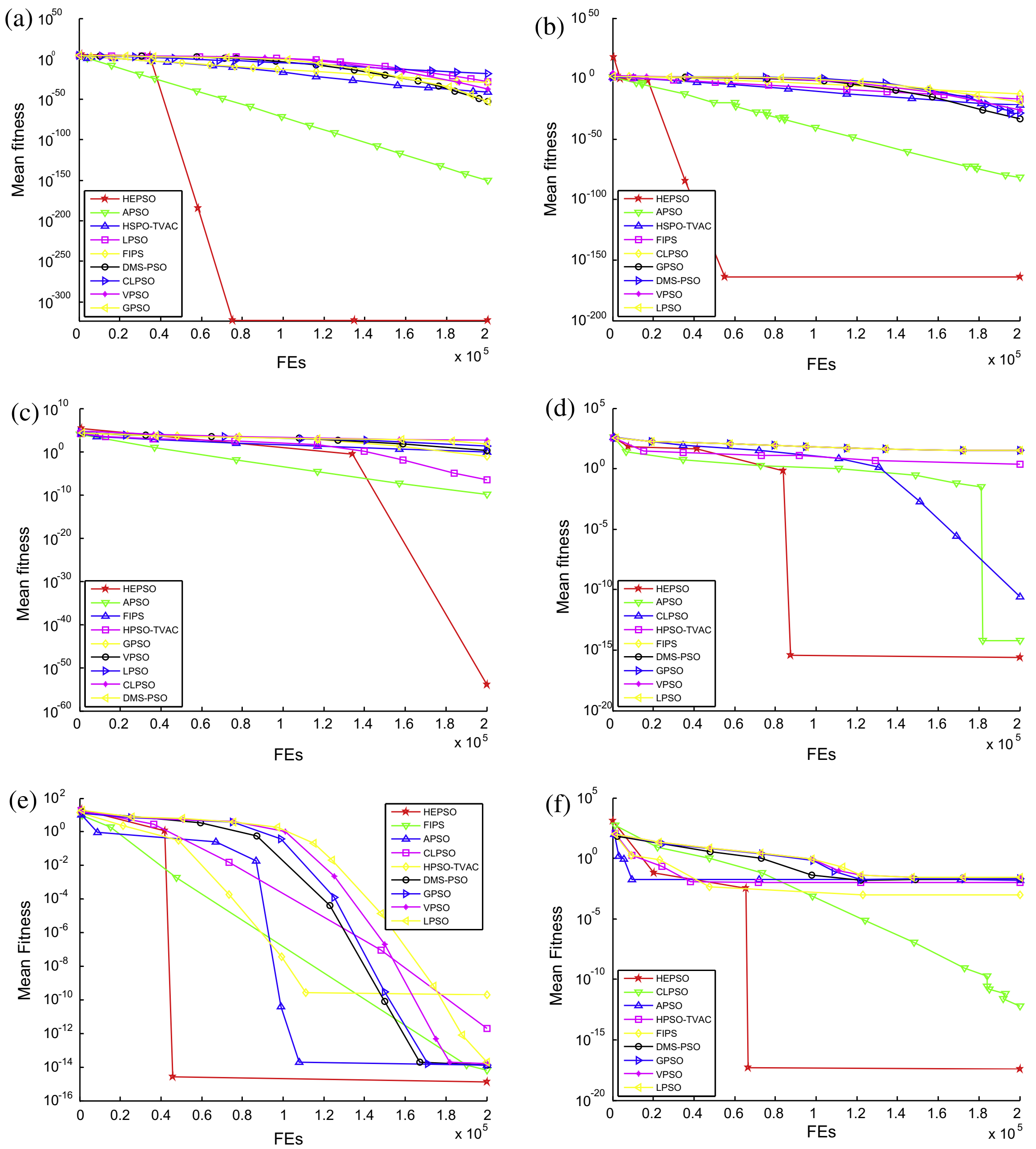

Fig. 5. The convergence trajectory of nine PSO algorithms on (a) Sphere, (b) Schwefel, (c) Quadric, (d) Rastrigin, (e) Ackley, and (f) Griwank functions.

The mean and standard deviation fitness of the best particles for thirty runs are summarized in Table 5. The population size, maximum iteration, and dimension are set at 20, 10,000, and 30, respectively. It can be observed from Table 5 that the proposed method has a very good performance. The convergence performance of the nine PSOs algorithms in the nine benchmark functions are shown in Fig. 5. Results in Table 5 and Fig. 5 illustrate the ability of HEPSO to find the global optimum in comparison with other well-known and recent algorithms.

Fig. 6 reinforces the results of Fig. 5 and Table 5, and shows the ability of HEPSO to achieve the global optimum for Rosenbrock function (the bold values indicate the best results). This figure shows when HEPSO finds the best particle, the algorithm continues searching to find the better answer; if it could not find a better answer, the old answer would be saved and used as global optimum.

A statistics comparison between HEPSO and six famous evolutionary algorithms (original PSO, SS-BLX [10], SS-Arith [10], DE-Exp [20], DE-Bin [20], SaDE [21]) on the six rotated functions listed in Table 2 is shown in Table 6. These test functions are

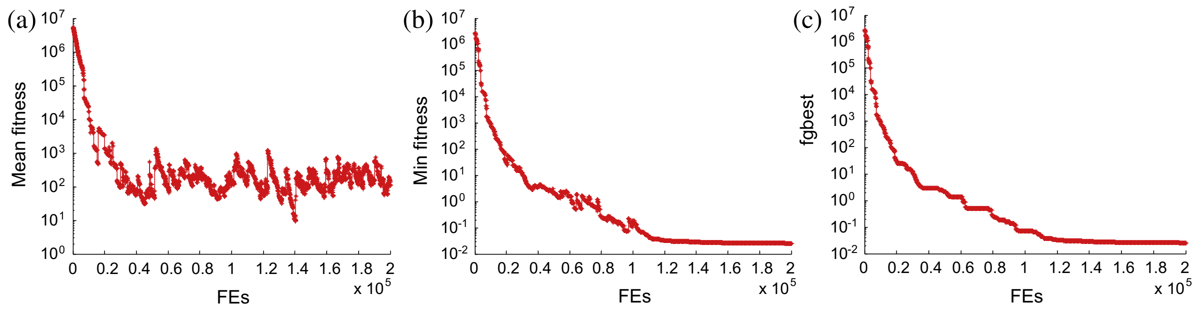

Fig. 6. The convergence trajectory of HEPSO on Rosenbrock function. (a) Mean fitness, (b) minimum fitness, and (c) best fitness in each iteration.

Table 6
Obtained results for the rotated benchmark functions.

Function	Original PSO	SS-BLX	SS-Arith	DE-Exp	DE-Bin	SaDE	HEPSO
f_{10}	$1.27 \times 10^{+03}$	3.40×10	1.06	8.26×10^{-9}	7.71×10^{-9}	8.41×10^{-9}	**0**
f_{11}	$6.61 \times 10^{+03}$	1.73	5.28	$\mathbf{8.18 \times 10^{-9}}$	8.34×10^{-9}	8.20×10^{-9}	4.02×10^{-04}
f_{12}	$1.32 \times 10^{+07}$	$1.14 \times 10^{+02}$	$4.94 \times 10^{+02}$	8.39×10^{-9}	$\mathbf{7.95 \times 10^{-9}}$	1.61×10^{-2}	4.86×10^{-03}
f_{13}	0.87×10	4.19	5.96	8.15×10^{-9}	4.54	8.33×10^{-9}	$\mathbf{2.37 \times 10^{-9}}$
f_{14}	2.13×10	$\mathbf{2.03 \times 10}$	$\mathbf{2.03 \times 10}$	$\mathbf{2.03 \times 10}$	$\mathbf{2.03 \times 10}$	$\mathbf{2.03 \times 10}$	2.14×10
f_{15}	2.81×10	$1.96 \times 10^{+03}$	$1.90 \times 10^{+03}$	$1.26 \times 10^{+03}$	$1.26 \times 10^{+03}$	$1.26 \times 10^{+03}$	$\mathbf{6.18 \times 10^{-01}}$

Note: The bold values indicate the best results.

Table 7
Results comparison of the algorithm convergence speed on nine benchmark functions listed in Table 1. '–' shows that the algorithm could not reach to acceptable solutions. FEs states the Function Evaluations.

Function		GPSO	LPSO	VPSO	DMS-PSO	CLPSO	HEPSO
f_1	Mean FEs	105,695	118,197	112,408	91,496	72,081	**18,204**
	Ratio%	100	100	100	100	100	**100**
f_2	Mean FEs	103,077	115,441	109,849	91,354	66,525	**15,067**
	Ratio%	100	100	100	100	100	**100**
f_3	Mean FEs	137,985	162,196	147,133	185,588	–	**78,011**
	Ratio%	100	96.7	100	86.7	0.0	**100**
f_4	Mean FEs	101,579	102,259	103,643	87,518	74,815	**16,115**
	Ratio%	100	100	100	100	100	**100**
f_5	Mean FEs	93,147	107,315	100,389	76,975	39,296	**30,000**
	Ratio%	100	100	100	100	100	**100**
f_6	Mean FEs	165,599	161,784	170,675	180,352	99,795	**12,292**
	Ratio%	80.0	26.7	43.3	40.0	100	**100**
f_7	Mean FEs	94,379	99,074	98,742	127,423	53,416	**7701**
	Ratio%	96.7	96.7	100	100	100	**100**
f_8	Mean FEs	110,844	125,543	118,926	100,000	76,646	**29,757**
	Ratio%	100	100	100	100	100	**100**
f_9	Mean FEs	111,733	125,777	117,946	97,213	81,422	**18,394**
	Ratio%	40.0	60.0	46.7	56.7	100	**100**
Mean reliability		90.7%	86.67%	87.7%	87.04%	88.88%	**100%**

Note: The bold values indicate the best results.

evaluated with 10 dimensions and 10 particles. All algorithms are independently run fifty times for each test function. Each run stops when the maximum number of evaluations 100,000 is reached. Results in Table 6 illustrate the capability of HEPSO to find the global optimum on the rotated test functions.

5.5. Analysis of the convergence speed of the algorithm

This section provides a statistics comparison on the convergence speed of the algorithms. The algorithms are executed in thirty independent runs and the mean number of function evaluations which is needed to the reach threshold (acceptable solution) is shown in Table 2. Thresholds are fixed, if the algorithm reaches these values, then it will exit from running and the accuracy speed of the algorithm is evaluated. Mean number of function evaluations (multiply maximum number of iterations by number of particles) needed to reach the acceptable solution are shown in Table 7. This table shows that HEPSO algorithm has a faster convergence speed than DMS-PSO [15], CLPSO [16], LPSO [13], VPSO [14], and GPSO [26]. Also, this table illustrates that HEPSO in comparison with others displays the highest percentage of reaching to the acceptable solutions on these test functions. For the mean reliability of all test functions, HEPSO exhibits the highest reliability of 100%.

6. Conclusion

In recent years, particle swarm optimization has appeared as a new and popular optimization algorithm due to its simplicity and efficiency. Nevertheless, according to the searching behavior of particle swarm optimization, this algorithm suffers from the premature convergence problem which results in a low optimization precision or even failure. To overcome this fault, in this paper, the HEPSO algorithm is introduced that is based on the combination of PSO with two new operators. The adaptive inertia weight and linear acceleration coefficients are implemented to help the particles that explore more areas in the solution space. Both un-rotated and rotated benchmark functions are applied to challenge the abilities of the proposed algorithm. Numerical results show that HEPSO performs very better in terms of convergence speed, global optimality, solution accuracy, and algorithm reliability, in comparison with well-known and recent evolutionary algorithms.

Reference

[1] B. Akay, D. Karaboga, A modified artificial bee colony algorithm for real-parameter optimization, Inf. Sci. 192 (2012) 120–142.
[2] P.J. Angeline, Using selection to improve particle swarm optimization, in: Proceedings of IEEE Congress on Evolutionary Computation, Anchorage, AK, 1998, pp. 84–89.
[3] A. Anler, A. Murat, R. Babu Chinnam, mr^{2}PSO: a maximum relevance minimum redundancy feature selection method based on swarm intelligence for support vector machine classification, Inf. Sci. 181 (2011) 4625–4641.
[4] W.D. Chang, A multi-crossover genetic approach to multivariable PID controllers tuning, Expert Syst. Appl. 33 (2007) 620–626.
[5] R.C. Eberhart, R. Dobbins, P.K. Simpson, Computational Intelligence PC Tools, Morgan Kaufmann Publishers, 1996.
[6] R.C. Eberhart, Y. Shi, Comparison between genetic algorithms and particle swarm optimization, in: Proceedings of the IEEE Congress on Evolutionary Computation, Anchorage, AK, May 1998, pp. 611–616.
[7] A.P. Engelbrecht, Computational Intelligence. An Introduction, John Wiley & Sons, 2002.
[8] A.P. Engelbrecht, Fundamentals of Computational Swarm Intelligence, John Wiley & Sons, 2005.
[9] Q. He, L. Wang, An effective co-evolutionary particle swarm optimization for constrained engineering design problems, Eng. Appl. Artif. Intell. 20 (2007) 89–99.
[10] F. Herrera, M. Lozano, D. Molina, Continuous scatter search: an analysis of the integration of some combination methods and improvement strategies, Eur. J. Oper. Res. 169 (2) (2006) 450–476.
[11] D. Karaboga, An Idea Based on Honey Bee Swarm For Numerical Optimization, Technical Report TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.
[12] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, vol. IV, 1995, pp. 1942–1948.
[13] J. Kennedy, R. Mendes, Population structure and particle swarm performance, in: Proceedings of the IEEE Congress on Evolutionary Computation, Honolulu, HI, 2002, pp. 1671–1676.
[14] J. Kennedy, R. Mendes, Neighborhood topologies in fully informed and best-of-neighborhood particle swarms, IEEE Trans. Syst. Man Cybern. Part C 36 (4) (2006) 515–519.
[15] J.J. Liang, P.N. Suganthan, Dynamic multi-swarm particle swarm optimizer, in: Proceedings of Swarm Intelligence Symposium, 2005, pp. 124–129.
[16] 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. 103 (2006) 281–295.
[17] M. Lovbjerg, T. K. Rasmussen, T. Krink, Hybrid particle swarm optimizer with breeding and subpopulations, in: Proceedings of Genetic, Evolutionary Computation, 2001, pp. 469–476.
[18] M.J. Mahmoodabadi, A. Bagheri, S. Arabani Mostaghim, M. Bisheban, Simulation of stability using Java application for Pareto design of controllers based on a new multi-objective particle swarm optimization, Math. Comput. Model. (2011) 1581–1607.
[19] R. Mendes, J. Kennedy, J. Neves, The fully informed particle swarm: simpler, maybe better, IEEE Trans. Evol. Comput. 83 (2004) 204–210.
[20] K.V. Price, M. Rainer, J.A. Lampinen, Differential Evolution: A Practical Approach to Global Optimization, Springer-Verlag, 2005.
[21] A.K. Qin, P.N. Suganthan, Self-adaptive differential evolution algorithm for numerical optimization, in: Proceedings of IEEE Congress on Evolutionary Computation, vol. 2, 2005, pp. 1785–1791.
[22] A. Ratnaweera, S.K. Halgamuge, Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficient, IEEE Trans. Evol. Comput. 83 (2004) 240–255.
[23] N. Ru, Y. Jian-hua, D. Shuai-qi, Hybrid particle swarm optimization-simplex algorithm for inverse problem, in: Proceedings of IEEE Conference on Chinese Control and Decision, 2010, pp. 3439–3442.
[24] M.H. Sadafi, R. Hosseini, H. Safikhani, A. Bagheri, M.J. Mahmoodabadi, Multi-Objective optimization of solar thermal energy storage using hybrid of particle swarm optimization, multiple crossover and mutation operator, Int. J. Eng. Trans. B 243 (2011) 367–376.
[25] T.D. Seeley, The Wisdom of the Hive: The Social Physiology of Honey Bee Colonies, Harvard University Press, 1995.
[26] Y. Shi, R.C. Eberhart, A modified particle swarm optimizer, in: Proceedings of the IEEE Congress on Evolutionary Computation, 1998, pp. 69–73.
[27] Y. Shi, Eberhart, Empirical study of particle swarm optimization, in: Proceedings of the Congress on Evolutionary Computation, Washington D.C., USA, 1999, pp. 1945–1940.
[28] S. Wang, J. Watada, A hybrid modified PSO approach to VaR-based facility location problems with variable capacity in fuzzy random uncertainty, Inf. Sci. 192 (2012) 3–18.
[29] H. Yoshida, K. Kawata, Y. Fukuyama, A particle swarm optimization for reactive power and voltage control considering voltage security assessment, IEEE Trans. Power Syst. 15 (2000) 1232–1239.
[30] E. Zahara, Y.T. Kao, Hybrid Nelder–Mead simplex search and particle swarm optimization for constrained engineering design problems, Expert Syst. Appl. 36 (2009) 3880–3886.
[31] Z. Zhan, J. Zhang, Y. Li, H.S. Chung, Adaptive particle swarm optimization, IEEE Trans. Syst. Man Cybern. Part B 39 (6) (2009) 1362–1381.