Int J Syst Assur Eng Manag (July-Sept 2015) 6(3):319–329
DOI 10.1007/s13198-015-0372-4

ORIGINAL ARTICLE

Fitness based particle swarm optimization

Kavita Sharma[1] · Varsha Chhamunya[2] · P C Gupta[3] · Harish Sharma[4] · Jagdish Chand Bansal[5]

Received: 7 May 2015 / Published online: 11 July 2015

Abstract Particle swarm optimization (PSO) is a popular population based approach used to solve nonlinear and complex optimization problems. It is simple to implement and swarm based probabilistic algorithm but, it also has drawbacks like it easily falls into local optima and suffers from slow convergence in the later stages. In order to reduce the chance of stagnation, while improving the convergence speed, a new position updating phase is incorporated with PSO, namely fitness based position updating in PSO. The proposed phase is inspired from the onlooker bee phase of Artificial Bee Colony (ABC) algorithm. In the proposed position updating phase, solutions update their positions based on probability which is a function of fitness. This strategy provides more position updating chances to the better solutions in the solution search process. The proposed algorithm is named as fitness based particle swarm optimization (FPSO). To show the efficiency of FPSO, it is compared with standard PSO 2011 and ABC algorithm over 15 well known benchmark problems and three real world engineering optimization problems.

Keywords Particle swarm optimization · Artificial Bee Colony · Swarm intelligence · Fitness based position updating · Optimization

✉ Harish Sharma
harish.sharma0107@gmail.com

Kavita Sharma
er_kavita28@yahoo.com

Varsha Chhamunya
varshagupta_28@yahoo.co.in

P C Gupta
dr.pcgupta@uok.ac.in

Jagdish Chand Bansal
jcbansal@gmail.com

[1] Government Polytechnic College, Kota, India

[2] Gurukul Institute of Engineering & Technology, Kota, India

[3] University of Kota, Kota, India

[4] Rajasthan Technical University, Kota, India

[5] South Asian University, New Delhi, India

1 Introduction

After being inspired from social behavior of fish schooling and birds flocking while searching the food, Kennedy and Eberhart (1995), Eberhart and Kennedy (1995) developed a swarm intelligence based optimization technique called particle swarm optimization (PSO) in 1995. PSO is a population based, easy to understand and implement, robust meta heuristic optimization algorithm. PSO can be a better choice for multi model, non convex, non linear and complex optimization problems but like any other evolutionary algorithm, it also has drawbacks like trapping into local optima (Liang et al. 2006), computationally inefficient as measured by the number of function evaluations required (Ciuprina et al. 2002). These points restrict PSO to less applicability (Li and Engelbrecht 2007). Researchers are continuously working to achieve these goals i.e., increasing convergence speed and ignoring the local optima to explore PSO applicability. As a result, a huge variants of PSO algorithm have been proposed (Ratnaweera et al. 2004; Liang et al. 2006; Zhan et al. 2009; Zhang et al. 2008; Gai-yun and Dong-xue 2009; Ciuprina et al. 2002) to get rid of these weaknesses. However,

achieving both goals simultaneously is difficult like liang et al. proposed the comprehensive-learning PSO (CLPSO) (Liang et al. 2006) which aims at ignoring the local optima, but results show that it also suffers from slow convergence. Ratnaweera et al. (2004) proposed time varying acceleration factors to balance cognitive and social component in initial and later stages. Zhan et al. (2009) also tried to adopt the acceleration factors increasing or decreasing depending on different exploring or exploiting search space stages. Zhang et al. (2008) studied effect of these factors on position expectation and variance and suggested that setting the cognitive acceleration factor as 1.85 and the social acceleration factor as 2 works good for improving system stability. Gai-yun and Dong-xue (2009) also worked for self adaption of cognitive and social factors. In this paper, a fitness based position update strategy is proposed in PSO to balance the exploration and exploitation capabilities. Further, the velocity update equation of PSO is also modified to improve the convergence ability.

Rest of the paper is organized as follows: Standard PSO is explained in Sect. 2. In Sect. 3, fitness based particle swarm optimization (FPSO) is proposed. In Sect. 4, performance of the proposed strategy is analyzed. Applications of FPSO to engineering optimization problems are explained in Sect. 5. Finally, in Sect. 6, paper is concluded.

2 Standard particle swarm optimization algorithm

PSO is an optimization technique which simulates the birds flocking behavior. PSO is a dynamic population of active, interactive agents with very little in the way of inherent intelligence. In PSO, whole group is called *swarm* and each individual is called *particle* which represents possible candidate's solution. The swarm finds food for its self through social learning by observing the behavior of nearby birds who appeared to be near the food source. Initially each particle is initialized within the search space randomly and keeps the information about its personal best position known as *pbest*, swarm best position known as *gbest* and current velocity *V* with which it is moving, in her memory. Based on these three values, each particle updates its position. In this manner, whole swarm moves in better direction while following collaborative trail and error method and converges to single best known solution.

For an D dimensional search space, the *i*th particle of the swarm is represented by a D- dimensional vector, $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$. The velocity of this particle is represented by another D-dimensional vector $V_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$. The previously best visited position of the *i*th particle is denoted as $P_i = (p_{i1}, p_{i2}, \ldots, p_{iD})$. g is the index of the best particle in the swarm. PSO swarm uses two equations for movement called *velocity update equation* and *position update equation*. The velocity of the *i*th particle is updated using the velocity update equation given by Eq. (1) and the position is updated using Eq. (2).

$$v_{ij} = v_{ij} + c_1 r_1 (p_{ij} - x_{ij}) + c_2 r_2 (p_{gj} - x_{ij}) \quad (1)$$

$$x_{ij} = x_{ij} + v_{ij} \quad (2)$$

where $j = 1, 2, \ldots, D$ represents the dimension and $i = 1, 2, \ldots, S$ represents the particle index. S is the size of the swarm and c_1 and c_2 are constants (usually $c_1 = c_2$), called cognitive and social scaling parameters respectively or simply acceleration coefficients. r_1 and r_2 are random numbers in the range [0,1] drawn from a uniform distribution.

The right hand side of velocity update equation (1) consists of three terms, the first term v_{ij} is the memory of the previous direction of movement which can be thought of as a momentum term and prevents the particle from drastically changing direction. The second term $c_1 r_1 (p_{ij} - x_{ij})$ is called cognitive component or persistence which draws particle back to their previous best situation and enables the local search in swarm. The last term $c_2 r_2 (p_{gj} - x_{ij})$ is known as social component which allows individuals to compare themselves to others in it's group and is responsible for global search. The pseudo-code for PSO, is described as follows:

Algorithm 1 Particle Swarm Optimization Algorithm:

```
Initialize the parameters, w, c1 and c2;
Initialize the particle positions and their velocities in the search space;
Evaluate fitness of individual particles;
Store gbest and pbest;
while stopping condition(s) not true do
   for each individual, Xi do
      for each dimension j, xij do
         (i) Evaluate the velocity vij using (1);
         (ii) Evaluate the position xij using (2);
      end for
   end for
   Evaluate fitness of updated particles;
   Update gbest and pbest;
end while
Return the individual with the best fitness as the solution;
```

Based on the neighborhood size, initially two versions of PSO algorithm were presented in literature namely, global version of PSO which is the original PSO (PSO-G) and the local version of PSO (PSO-L) (Kennedy and Eberhart 1995). The only difference between PSO-G and PSO-L is that the term p_g in social component in velocity update equation (1). For PSO-G, it refers the best particle of whole swarm while for PSO-L it represents the best particle of the individual's neighborhood. The social network employed by the PSO-G reflects the star topology which offers a faster convergence but it is very likely to converge prematurely. While PSO-L uses a ring social network topology where smaller neighborhoods are defined for each particle. It can be easily observed that due to the less particle inter connectivity in PSO-L, it is less susceptible to be trapped in local minima but at the cost of slow convergence. In general, PSO-G performs better for unimodal problems and PSO-L for multimodal problems.

Velocity update equation in PSO determines the balance between exploration and exploitation capability of PSO. In Basic PSO, no bounds were defined for velocity, due to which in early iterations the particles far from gbest, will take large step size and are very much intended to leave the search space. Thus to control velocity so that particle update step size is balanced, velocity clamping concept was introduced. In velocity clamping, whenever velocity exceeds from its bounds, it is set at its bounds. To avoid the use of velocity clamping and to make balance between exploration and exploitation, a new parameters called inertia weight (Shi and Eberhart 1998) was introduced in velocity update equation as:

$$v_{ij} = w \times v_{ij} + c_1 r_1 (p_{ij} - x_{ij}) + c_2 r_2 (p_{gj} - x_{ij}) \tag{3}$$

where inertia weight is denoted by w. In subsequent section, the proposed PSO algorithm is explained in details.

3 Fitness based particle swarm optimization

However the standard PSO has the capability to get a good solution at a significantly faster rate but, when it is compared to other optimization techniques, it is weak to refine the optimum solution, mainly due to less diversity in later search (Angeline 1998). On the different side, problem-based tuning of parameters is also important in PSO, to get optimum solution accurately and efficiently (Shi and Eberhart 1998). In standard PSO velocity update equation (1) contains three terms. The first term has the global search capability, the second and third terms are the particles cognitive and social information sharing capability respectively. More cognitive capability force particle to move towards personal best position fast and more social information force particle to move towards global best position fast. It can be seen from (1), the movement of swarm towards optimum solution is guided by the acceleration factor c_1 and c_2. Therefore, acceleration coefficient c_1 and c_2 should be tuned carefully to get the desired solution.

Kennedy and Eberhart (1995) explained that more value of the cognitive component compared to the social component, results in excessive wandering of individuals through the search space while on the other hand more value of the social component may results that particles will converge prematurely toward a local optimum. These two component play important role for balancing the exploration and exploitation capabilities of PSO. Therefore, in this paper two modifications are proposed for improving the solution search efficiency of PSO.

1. Velocity update equation (refer Eq. 3) of PSO is modified as follows:

$$v_{ij} = w \times v_{ij} + c \times r(p_{gj} - x_{ij}) \tag{4}$$

   It is clear from Eq. (4) that the *Pbest* component (cognitive component) is removed $\{c_1 r_1 (p_{ij} - x_{ij})\}$ from the velocity update equation of PSO. Now the magnitude of velocity of each individual will depend on its distance from the current global best solution. Therefore, this strategy will improve the exploitation capability of PSO.
2. A new position update process, which is inspired from the Artificial Bee Colony (ABC) algorithm's onlooker bee phase (Karaboga and Basturk 2007) is incorporated with PSO. In employed bee phase of ABC, all the employed bees search the food source and calculate their fitness using Eq. (5):

$$fitness_i = \begin{cases} 1/(1+f_i), & \text{if } f_i \geq 0 \\ 1 + abs(f_i), & \text{if } f_i < 0. \end{cases} \tag{5}$$

and then in the onlooker bee phase, onlooker bees analyze the available information and select a solution with a probability, $prob_i$, related to its fitness. The probability $prob_i$ may be calculated using Eq. (6):

$$prob_i(G) = \frac{0.9 \times fitness_i(G)}{maxfit(G)} + 0.1, \tag{6}$$

where G is the iteration counter, $fitness_i(G)$ is the fitness value of ith solution and $maxfit(G)$ is the maximum fitness of the solutions in Gth iteration. Position update equation of ABC is shown in Eq. (7):

$$y_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \tag{7}$$

where $k \in \{1, 2, \ldots, S\}$ and $j \in \{1, 2, \ldots, D\}$ are randomly chosen indices, k must be different from i, ϕ_{ij} is a random number between $[-1, 1]$ and x_{kj} is a random individual in the current population. In the basic ABC, at any given time,

only one dimension is updated in employed or onlooker bee phase. In onlooker bee phase this update takes place based on a probability which is a function of fitness.

The proposed position update strategy is incorporated with PSO and the newly developed algorithm is named as FPSO. In FPSO, Algorithm 2 is applied after basic PSO operators. The insertion of Algorithm 2 makes FPSO more capable of exploitation in the better search regions. It is expected because in FPSO after applying basic PSO operators, better candidate solutions are offered more chances to update themselves than worse candidates. The pseudo-code of the proposed position update strategy which works after PSO operators is shown in Algorithm 2.

Algorithm 2 Fitness based Position Update Phase:

for each individual, x_i **do**
  if $prob_i > rand(0,1)$ **then**
    $y_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj})$,
    Calculate fitness of $\mathbf{y_i}$,
    Apply greedy selection between $\mathbf{y_i}$ and $\mathbf{x_i}$,
  end if
end for

The pseudo-code for the proposed FPSO algorithm is shown in Algorithm 3.

Algorithm 3 Fitness based Particle Swarm Optimization(FPSO):

Initialize the parameters, w, and c and S;
Initialize the particle positions and their velocities in the search space;
Evaluate fitness of individual particles;
Store the gbest solution;
while stopping condition(s) not true **do**
  for each individual, X_i **do**
    for each dimension j of x_{ij} **do**
      (i) Evaluate the velocity v_{ij} using (4);
      (ii) Evaluate the position x_{ij} using (2);
    end for
  end for
  Evaluate fitness of updated particles;
  Update gbest solution;
  /******Fitness based position update phase in FPSO **********/
  $t = 1, i = 1$ /*** t counts number of updates ***/
  while $t \leq S$ **do**
    if $prob_i > rand(0,1)$ **then**
      /****** $prob_i$ is the probability of an individual x_i described by equation (6)****/
      $y_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj})$,
      $\{k, j$ is randomly selected index$\}$
      Calculate fitness of $\mathbf{y_i}$;
      Apply greedy selection between $\mathbf{y_i}$ and $\mathbf{x_i}$.;
      $t = t + 1$
    end if
    $i = i + 1$
    if $i > S$ **then**
      $i = 1$
    end if
  end while
end while
Return the individual with the best fitness as the solution;

4 Experiments and results

This section shows performance evaluation of the proposed algorithm in terms of accuracy, efficiency and reliability.

4.1 Test problems under consideration

To validate the effectiveness of FPSO, 15 mathematical optimization problems (f_1–f_{15}) of different characteristics and complexities are taken into consideration (listed in Table 1). These all problems are continuous in nature.

4.2 Experimental setting

The results obtained from the proposed FPSO are stored in the form of success rate, average number of function evaluations, and mean error. Results for these test problems (Table 1) are also obtained from ABC and PSO for the comparison purpose. The following parameter setting is adopted while implementing the proposed and other considered algorithms to solve the problems:

Parameter setting for PSO (Standard PSO 2011) and FPSO:

– Swarm size $S = 50$,

Table 1 Test problems

Objective function	Search range	Optimum value	D	AE
$f_1(x) = \sum_{i=1}^{D}(100(x_{i+1} - x_i{}^2)^2 + (x_i - 1)^2)$	[−30, 30]	$f(\mathbf{1}) = \mathbf{0}$	30	1.0E−02
$f_2(x) = 10D + \sum_{i=1}^{D}[x_i^2 - 10\cos(2\pi x_i)]$	[−5.12, 5.12]	$f(\mathbf{0}) = \mathbf{0}$	30	1.0E−05
$f_3(x) = -\sum_{i=1}^{D} \sin x_i(\sin(\frac{ix_i{}^2}{\pi})^{20})$	$[0, \pi]$	$f_{min} = -9.66015$	10	1.0E−05
$f_4(x) = \sum_{i=1}^{D} x_i{}^2 + (\sum_{i=1}^{D}\frac{ix_i}{2})^2 + (\sum_{i=1}^{D}\frac{ix_1}{2})^4$	[−5.12, 5.12]	$f(\mathbf{0}) = \mathbf{0}$	30	1.0E−02
$f_5(x) = \sum_{i=1}^{D} ix_i^4 + \text{ random } [0,1)$	[−1.28, 1.28]	$f(\mathbf{0}) = \mathbf{0}$	30	1.0E−05
$f_6(x) = -\sum_{i=1}^{D-1}\left(\exp\left(\frac{-(x_i^2+x_{i+1}^2+0.5x_ix_{i+1})}{8}\right) \times \mathrm{I}\right)$ where, $\mathrm{I} = cos\left(4\sqrt{x_i^2 + x_{i+1}^2 + 0.5x_ix_{i+1}}\right)$	[−5, 5]	$f(\mathbf{0}) = -\mathbf{D} + \mathbf{1}$	10	1.0E−05
$f_7(x) = \sum_{i=1}^{D}(x_i - 1)^2 - \sum_{i=2}^{D} x_ix_{i-1}$	$[-D^2, D^2]$	$f_{min} = -\frac{(D(D+4)(D-1))}{6}$	10	1.0E−01
$f_8(x) = 100[x_2 - x_1^2]^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 + 10.1[(x_2 - 1)^2 + (x_4 - 1)^2] + 19.8(x_2 - 1)(x_4 - 1)$	[−10, 10]	$f(\mathbf{1}) = \mathbf{0}$	4	1.0E−05
$f_9(x) = \sum_{i=1}^{11}[a_i - \frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}]^2$	[−5, 5]	$f(0.192833,\ 0.190836, 0.123117, 0.135766) = 3.075E{-}04$	4	1.0E−05
$f_{10}(x) = \sum_{i=1}^{D-1}(100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + f_{bias}$, $z = x - o + 1\ \ x = [x_1, x_2, \ldots, x_D],\ o = [o_1, o_2, \ldots, o_D]$	[−100, 100]	$f(o) = f_{bias} = 390$	10	1.0E−01
$f_{11}(x) = (1 + (x_1 + x_2 + 1)^2 \cdot (19{-}14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)) \cdot (30 + (2x_1 - 3x_2)^2 \cdot (18{-}32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2))$	[−2, 2]	$f(0, -1) = 3$	2	1.0E−14
$f_{12}(x) = -\cos x_1 \cos x_2 e^{((-(x_1-\pi)^2-(x_2-\pi)^2))}$	[−10, 10]	$f(\pi, \pi) = -1$	2	1.0E−13
$f_{13}(x) = \sum_{i=1}^{5}(\frac{x_1x_3t_i}{1+x_1t_i+x_2v_i} - y_i)^2$	[−10, 10]	$f(3.13,\ 15.16, 0.78) = 0.4E{-}04$	3	1.0E−03
$f_{14}(x) = -\sum_{i=1}^{5} i\cos((i+1)x_1 + 1)\sum_{i=1}^{5} i\cos((i+1)x_2 + 1)$	[−10, 10]	$f(7.0835, 4.8580) = -186.7309$	2	1.0E−05
$f_{15}(x) = -[A\prod_{i=1}^{D}\sin(x_i - z) + \prod_{i=1}^{D} sin(B(x_i - z))]$, $A = 2.5, B = 5, z = 30$	[0, 180]	$f(\mathbf{90} + \mathbf{z}) = -(\mathbf{A} + \mathbf{1})$	10	1.0E−02

AE acceptable error

Table 2 Comparison based on average number of function evaluations

TP	ABC	PSO	FPSO
f_1	196,024	199,407	193,681
f_2	49,984	200,050	102,911
f_3	29,192	198,326	69,919
f_4	200,000	196,434	125,547
f_5	200,000	200,000	200,000
f_6	89,944	195,748	67,107
f_7	200,000	67,527	44,941
f_8	200,000	52,689	21,049
f_9	172,839	35,314	15,399
f_{10}	174,699	187,126	67,504
f_{11}	112,961	102,884	87,733
f_{12}	187,003	9818	10,599
f_{13}	28,346	3397	2926
f_{14}	4861	80,823	9836
f_{15}	55,076	177,156	104,254

TP test problem

Table 3 Comparison based on success rate out of 100 runs

TP	ABC	PSO	FPSO
f_1	5	1	23
f_2	100	0	100
f_3	100	3	100
f_4	0	31	100
f_5	0	0	0
f_6	97	6	100
f_7	0	100	100
f_8	0	100	100
f_9	21	100	100
f_{10}	23	58	98
f_{11}	60	51	59
f_{12}	13	100	100
f_{13}	100	100	100
f_{14}	100	73	100
f_{15}	99	27	99

TP test problem

- Inertia weight $w = 0.8$,
- Acceleration coefficients $c = c_1 = c_2 = 0.5 + log2$ (for PSO) (Kim et al. 2009),
- The number of run =100.
- The terminating criteria: Either acceptable error (Table 1) meets or maximum number of function evaluations (which is set to be 200000) is reached,

Parameter setting for ABC:

- Colony size $S = 50$ (Diwold et al. 2011; El-Abd 2011),
- $\phi_{ij} = rand[-1, 1]$,
- Number of food sources $SN = S/2$,
- The number of run = 100,
- The terminating criteria: Either acceptable error (Table 1) meets or maximum number of function evaluations (which is set to be 200000) is reached,

Table 4 Comparison based on mean error

TP	ABC	PSO	FPSO
f_1	1.34E+00	4.44E+01	6.80E+00
f_2	5.66E−06	3.87E+01	7.65E−06
f_3	3.84E−06	3.12E−01	5.04E−06
f_4	9.75E+01	2.20E−02	9.72E−03
f_5	1.18E+01	9.94E+00	8.95E+00
f_6	1.05E−02	1.48E+00	8.64E−06
f_7	8.89E−01	9.53E−06	8.98E−06
f_8	1.52E−01	8.17E−04	7.65E−04
f_9	1.69E−04	8.96E−05	8.21E−05
f_{10}	6.26E−01	1.69E+00	1.66E−01
f_{11}	1.19E−06	4.96E−14	4.22E−14
f_{12}	2.44E−05	5.34E−14	5.10E−14
f_{13}	1.95E−03	1.95E−03	1.88E−03
f_{14}	4.58E−06	9.54E−05	5.42E−06
f_{15}	7.89E−03	3.74E−01	7.81E−03

TP test problem

4.3 Results analysis of experiments

Tables 2, 3 and 4 present the numerical results for the benchmark problems of Table 1 with the experimental settings shown in Sect. 4.2. These tables show the results of the proposed and other considered algorithms in terms of average number of function evaluations (AFE), success rate (SR) and mean error (ME).

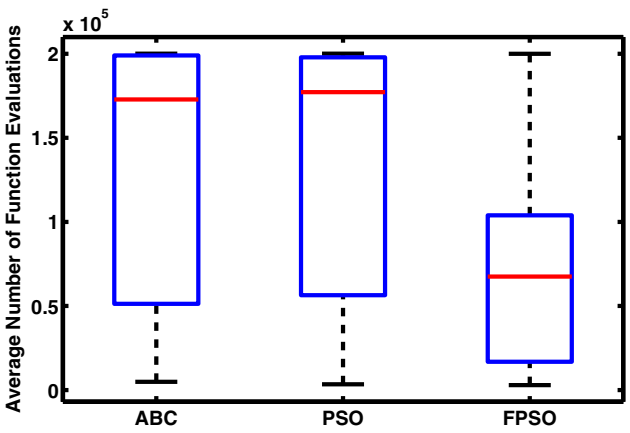


Fig. 1 Boxplots graph for average number of function evaluation

Table 5 Comparison based on mean function evaluations and the Mann–Whitney U rank sum test at a $\alpha = 0.05$ significance level ('+' indicates FPSO is significantly better, '−' indicates FPSO is worse and '=' indicates that there is no significant difference)

TP	Mann–Whitney U rank sum test with FPSO		TP	Mann–Whitney U rank sum test with FPSO	
	ABC	PSO		ABC	PSO
f_1	+	+	f_9	+	+
f_2	−	+	f_{10}	+	+
f_3	−	+	f_{11}	+	+
f_4	+	+	f_{12}	+	−
f_5	=	=	f_{13}	+	+
f_6	+	+	f_{14}	−	+
f_7	+	+	f_{15}	−	+
f_8	+	+			

TP test problem

After analyzing the results, it can be said that FPSO outperforms the considered algorithms most of the time in terms of accuracy (due to ME), reliability (due to SR) and efficiency (due to AFE). Some other statistical tests like the Mann–Whitney U rank sum test, acceleration rate (AR) (Rahnamayan et al. 2008) and boxplots have also been done in order to analyze the algorithms output more intensively.

4.4 Statistical analysis

Algorithms ABC, PSO and FPSO are compared based on SR, AFE, and ME. From the results shown in Table 2, it is clear that FPSO costs less on 9 test functions (f_1,f_4,f_6,f_7,f_8,f_9,f_{10},f_{11},f_{13}) among all the considered algorithms. As these functions include unimodel, multimodel, separable, non separable, lower and higher dimension functions, it can be stated that FPSO balances the exploration and exploitation capabilities efficiently for all kind of functions compared to the other considered algorithms. ABC outperforms FPSO over test functions f_2,f_3,f_{14},f_{15}, while PSO is outperforms over FPSO on test functions f_{12}.

Since boxplots (Williamson et al. 1989) can efficiently represent the empirical distribution of results, the boxplots for average number of function evaluations for PSO, ABC and FPSO have been represented in Fig. 1. Figure 1 shows that FPSO is cost effective in terms of function evaluations as interquartile range and median of average number of function evaluations are very low for FPSO.

Though, it is clear from box plots that FPSO is cost effective than ABC and PSO i.e., FPSO's result differs from the other, now to check, whether there exists any significant difference between algorithm's output or this difference is due to some randomness, we require another statistical test. It can be observed from boxplots of Fig. 1 that average number of function evaluations used by the considered algorithms to solve the different problems are not normally distributed, so a non-parametric statistical test is required to compare the performance of the algorithms. The Mann–Whitney U rank sum (Mann and Whitney 1947), a non-parametric test, is well established test for comparison among non-Gaussian data. In this paper, this test is performed at 5% level of significance ($\alpha = 0.05$) between FPSO - ABC and FPSO - PSO.

Table 5 shows the results of the Mann–Whitney U rank sum test for the average number of function evaluations of 100 simulations. First we observe the significant difference by Mann–Whitney U rank sum test i.e., whether the two data sets are significantly different or not. If significant difference is not seen (i.e., the null hypothesis is accepted) then sign '=' appears and when significant difference is observed i.e., the null hypothesis is rejected then compare

Table 6 Acceleration rate (AR) of FPSO as compared to ABC and PSO

TP	ABC	PSO
f_1	1.01209718	1.029561419
f_2	0.485698877	1.943903257
f_3	0.417512568	2.836490536
f_4	1.593022561	1.564618969
f_5	1.00019985	1.00025
f_6	1.340312037	2.916931789
f_7	4.450313185	1.502564445
f_8	9.502562531	2.50312359
f_9	11.22408988	2.293298266
f_{10}	2.587992119	2.772072766
f_{11}	1.287559869	1.172694425
f_{12}	17.64270956	0.926317279
f_{13}	9.68765892	1.160970608
f_{14}	0.494260878	8.217110614
f_{15}	0.528288507	1.69927293

TP test problem

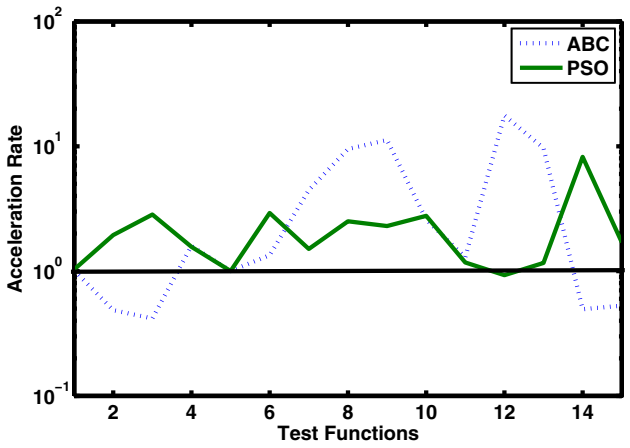


Fig. 2 Acceleration rate of FPSO as compared to ABC and PSO

the average number of function evaluations. And we use signs '+' and '−' for the case where FPSO takes less or more average number of function evaluations than the other algorithms, respectively. Therefore in Table 5, '+' shows that FPSO is significantly better and '−' shows that FPSO is significantly worse. As Table 5 includes 23 '+' signs out of 30 comparisons. Therefore, it can be concluded that the results of FPSO is significantly cost effective than ABC and PSO over considered test problems.

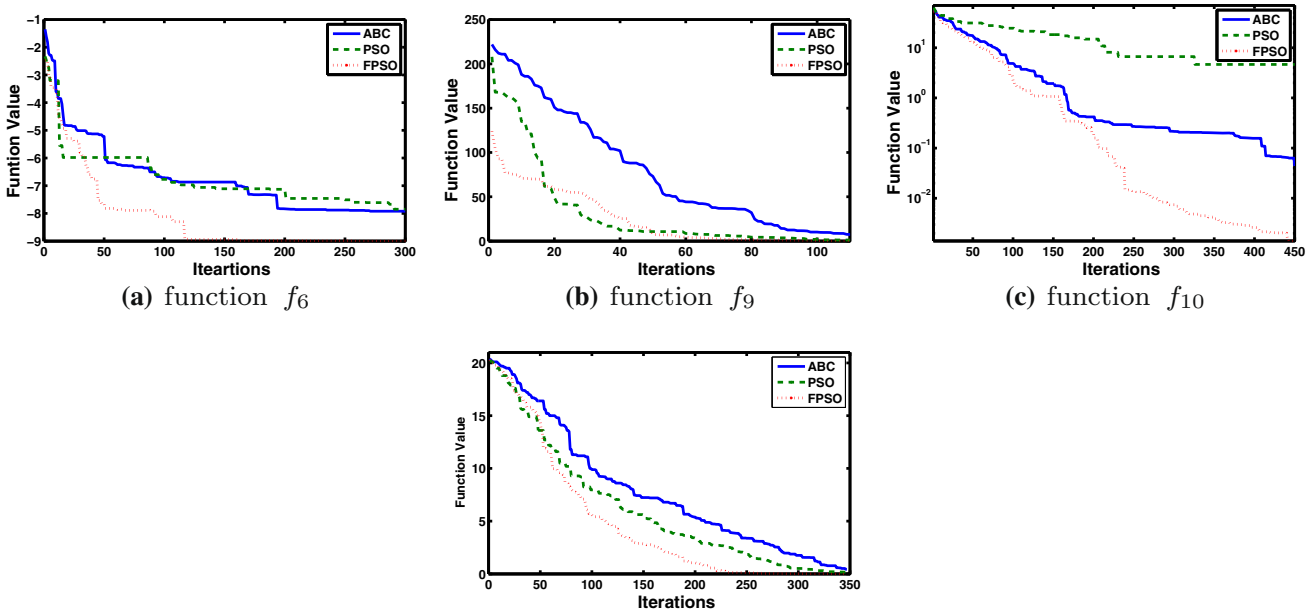
Further, we compare the convergence speed of the considered algorithms by measuring the AFEs. A smaller AFEs means higher convergence speed. In order to minimize the effect of the stochastic nature of the algorithms, the reported function evaluations for each test problem is averaged over 100 runs. In order to compare convergence speeds, we use the acceleration rate (AR) which is defined as follows, based on the AFEs for the two algorithms ALGO and FPSO:

$$AR = \frac{AFE_{ALGO}}{AFE_{FPSO}}, \tag{8}$$

where, $ALGO \in \{\text{ABC}, \text{PSO}\}$ and $AR > 1$ means FPSO is faster. In order to investigate the AR of the proposed algorithm as compare to the considered algorithms, results of Table 2 are analyzed and the value of AR is calculated using Eq. (8). Table 6 shows a comparison between FPSO and ABC, FPSO and PSO in terms of AR. It is clear from the Table 6 that convergence speed of FPSO is better than considered algorithms for most of the functions. The same claim can also be justified by visualizing Fig. 2. The convergence speed of FPSO can also be judged through convergence Fig. 3a–d that show the fitness movement through the iterations in a single run for selective functions f_6, f_9, f_{10} and f_{11} 3 respectively.

5 Applications of FPSO to engineering optimization problems

Further, the proposed FPSO algorithm is applied to solve three real world engineering optimization problems, namely compression spring (Onwubolu and Babu 2004; Sandgren 1990), Lennard–Jones (Clerc 2012), and Welded



(a) function f_6 **(b)** function f_9 **(c)** function f_{10}

(d) function f_{11}

Fig. 3 Convergence graph for ABC, PSO and FPSO on test problems f_6, f_9, f_{10}, f_{11}

Table 7 Comparison of the results of test problems

TP	Algorithm	SD	ME	AFE	SR
E_1	ABC	1.17E−02	1.36E−02	187,602.32	10
	PSO	2.99E−04	4.86E−04	23,789.5	100
	FPSO	9.09E−04	5.30E−04	18,159	100
E_2	ABC	1.16E−04	8.68E−04	71,931.02	100
	PSO	3.38E−01	1.95E−01	139,263.5	53
	FPSO	1.48E−04	8.41E−04	62,669.5	100
E_3	ABC	7.65E−02	2.40E−01	196,524	3
	PSO	4.29E−03	9.43E−02	4012.5	100
	FPSO	5.61E−03	9.36E−02	4826	100

TP test problem

beam design optimization (Ragsdell and Phillips 1976; Mahdavi et al. 2007). Each of the engineering optimization problem is described as follows:

5.1 Compression spring

This problem minimizes the weight of a compression spring, subject to constraints of minimum deflection, shear stress, surge frequency, and limits on outside diameter and on design variables. There are three design variables: the wire diameter x_1, the mean coil diameter x_2, and the number of active coils x_3. This is a simplified version of a more difficult problem. The mathematical formulation of this problem is:

$x_1 \in \{1, \ldots, 70\}$ granularity 1
$x_2 \in [0.6; 3]$
$x_3 \in [0.207; 0.5]$ granularity 0.001

and four constraints

$$g_1 := \frac{8C_f F_{max} x_2}{\pi x_3^3} - S \leq 0$$
$$g_2 := l_f - l_{max} \leq 0$$
$$g_3 := \sigma_p - \sigma_{pm} \leq 0$$
$$g_4 := \sigma_w - \frac{F_max - F_p}{K} \leq 0$$

with

$$C_f = 1 + 0.75\frac{x_3}{x_2 - x_3} + 0.615\frac{x_3}{x_2}$$
$$F_{max} = 1000$$
$$S = 189{,}000$$
$$l_f = \frac{F_{max}}{K} + 1.05(x_1 + 2)x_3$$
$$l_{max} = 14$$
$$\sigma_p = \frac{F_p}{K}$$
$$\sigma_{pm} = 6$$
$$F_p = 300$$
$$K = 11.5 \times 10^6 \frac{x_3^4}{8x_1x_2^3}$$
$$\sigma_w = 1.25$$

and the function to be minimized is

$$E_1(\mathbf{X}) = \pi^2 \frac{\mathbf{x_2 x_3^2 (x_1 + 2)}}{\mathbf{4}}$$

The best known solution is (7, 1.386599591, 0.292) , which gives the fitness value $f^* = 2.6254$. Acceptable error for this problem is 1.0E−04.

5.2 Lennard–Jones

The function to minimize is a kind of potential energy of a set of N atoms. The position X_i of the atom i has three coordinates, and therefore the dimension of the search space is $3N$. In practice, the coordinates of a point X are the concatenation of the ones of the X_i. In short, we can write $X = (X_1, X_2, \ldots, X_N)$, and we have then

$$E_2(\mathbf{X}) = \sum_{\mathbf{i=1}}^{\mathbf{N-1}} \sum_{\mathbf{j=i+1}}^{\mathbf{N}} \left(\frac{\mathbf{1}}{\|\mathbf{X_i} - \mathbf{X_j}\|^{\mathbf{2\alpha}}} - \frac{\mathbf{1}}{\|\mathbf{X_i} - \mathbf{X_j}\|^{\alpha}} \right)$$

In this study $N = 5, \alpha = 6$, and the search space is $[-2, 2]$ (Clerc 2012).

5.3 Welded beam design optimization problem

The problem is to design a welded beam for minimum cost, subject to some constraints (Ragsdell and Phillips 1976; Mahdavi et al. 2007). The objective is to find the minimum fabricating cost of the welded beam subject to constraints on shear stress τ, bending stress σ, buckling load P_c, end deflection δ, and side constraint. There are four design variables: x_1, x_2, x_3 and x_4. The mathematical formulation of the objective function is described as follows:

$$E_3(\mathbf{x}) = \mathbf{1.10471 x_1^2 x_2 + 0.04811 x_3 x_4 (14.0 + x_2)}$$

subject to:

Table 8 Comparison based on mean function evaluations and the Mann–Whitney U rank sum test at a $\alpha = 0.05$ significance level ('+' indicates FPSO is significantly better)

Function	FPSO vs PSO	FPSO vs ABC
E_1	+	+
E_2	+	+
E_3	+	+

TP test problem

$$g_1(\mathbf{x}) = \tau(\mathbf{x}) - \tau_{\mathbf{max}} \leq \mathbf{0}$$
$$g_2(\mathbf{x}) = \sigma(\mathbf{x}) - \sigma_{\mathbf{max}} \leq \mathbf{0}$$
$$g_3(\mathbf{x}) = x_1 - x_4 \leq 0$$
$$g_4(\mathbf{x}) = \delta(\mathbf{x}) - \delta_{\mathbf{max}} \leq \mathbf{0}$$
$$g_5(\mathbf{x}) = P - P_c(\mathbf{x}) \leq \mathbf{0}$$

$0.125 \leq x_1 \leq 5, \quad 0.1 \leq x_2, x_3 \leq 10$ and $0.1 \leq x_4 \leq 5$

where

$$\tau(\mathbf{x}) = \sqrt{\tau'^{\mathbf{2}} - \tau'\tau''\frac{\mathbf{x_2}}{\mathbf{R}} + \tau''^{\mathbf{2}}},$$

$$\tau' = \frac{P}{\sqrt{2}x_1x_2}, \tau'' = \frac{MR}{J}, M = P(L + \frac{x_2}{2}),$$

$$R = \sqrt{\frac{{x_2}^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2},$$

$$J = 2/\left(\sqrt{2}x_1x_2\left[\frac{{x_2}^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right),$$

$$\sigma(\mathbf{x}) = \frac{\mathbf{6PL}}{\mathbf{x_4x_3^2}}, \delta(\mathbf{x}) = \frac{\mathbf{6PL^3}}{\mathbf{Ex_4x_3^2}},$$

$$P_c(\mathbf{x}) = \frac{\mathbf{4.013Ex_3x_4^3}}{\mathbf{6L^2}}\left(\mathbf{1} - \frac{\mathbf{x_3}}{\mathbf{2L}}\sqrt{\frac{\mathbf{E}}{\mathbf{4G}}}\right),$$

$P = 6000\,\text{lb}, \ L = 14\,\text{in.}, \ \delta_{max} = 0.25\,\text{in.},$
$\sigma_{max} = 30{,}000\,\text{psi},$

$\tau_{max} = 13{,}600\,\text{psi}, \ E = 30 \times 10^6\,\text{psi}, \ G = 12 \times 10^6\,\text{psi}.$

The best known solution is (0.205730, 3.470489, 9.036624, 0.205729) , which gives the function value 1.724852. Acceptable error for this problem is 1.0E−01.

5.4 Experimental results

To solve the constraint optimization problems (E_1 and E_3), a penalty function approach is used in the experiments. In this approach the search is modified by converting the original problem into an unconstrained optimization problem by adding a penalty term in case of constraints violation as shown below:

$$f(x) = f(x) + \beta$$

where, $f(x)$ is the original function value and β is the penalty term which is set to 10^3.

Table 7 shows the experimental results of the considered algorithms on the engineering optimization problems. It is clear from Table 7 that the inclusion of new position update strategy in the PSO 2011, it performs better than the considered algorithms.

Further, the algorithms are compared through *SR*, *ME* and *AFE*. On the basis of results shown in Table 7, it can be stated that the FPSO performs better than the PSO 2011 and ABC algorithms for the considered engineering problems.

Further, the Mann–Whitney U rank sum test as mentioned in Sect. 4.4 is applied to the considered algorithms for the average number of function evaluations of 100 simulations. The result of the test is shown in Table 8. It is clear from Table 8 that FPSO is an efficient algorithms in the same category.

6 Conclusion

In this paper, inspired from onlooker bee phase of ABC algorithm, a new solution search phase is introduced in PSO. In the proposed phase, solutions get chance to update themselves on the basis of probability which is a function of fitness. In this way, the high fit solutions get more chance to update themselves as compared to low fit solutions. Further, velocity update equation of PSO is also modified and the previous best learning component (cognitive component) is deleted from the velocity update equation. This modification helps the solutions to converge fast to the global optima. Through the experiments on 15 complex well known benchmark functions and three real world engineering optimization problems, it is shown that the proposed strategy is a competitive candidate in the field of swarm intelligence based algorithms.

References

Angeline P (1998) Evolutionary optimization versus particle swarm optimization: philosophy and performance differences. In: Androutsos D, Plataniotis KN, Venetsanopoulos AN (eds) Evolutionary programming VII. Springer, Berlin, pp 601–610

Ciuprina G, Ioan D, Munteanu I (2002) Use of intelligent-particle swarm optimization in electromagnetics. IEEE Trans Magn 38(2):1037–1040

Clerc M (2012) List based pso for real problems

Diwold K, Aderhold A, Scheidler A (2011) Performance evaluation of artificial bee colony optimization and new selection schemes. Memetic Comput 3(3):1–14

Eberhart R, Kennedy J (1995) A new optimizer using particle swarm theory. In: Proceedings of the sixth international symposium on micro machine and human science (MHS'95). IEEE, Nagoya, pp 39–43

El-Abd M (2011) Performance assessment of foraging algorithms vs. evolutionary algorithms. Inf Sci 182:243–263

Gai-yun W, Dong-xue H (2009) Particle swarm optimization based on self-adaptive acceleration factors. In: 3rd international conference on genetic and evolutionary computing 2009 (WGEC'09). IEEE, Danvers, pp 637–640

Karaboga D, Basturk B (2007) Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems. Found Fuzzy Logic Soft Comput 4529(1):789–798

Kennedy J, Eberhart (1995a) Particle swarm optimization. In: Proceedings of IEEE international conference on neural networks, vol 4. IEEE Service Center, Perth, pp 1942–1948

Kennedy J, Eberhart RC (1995b) A new optimizer using particle swarm theory. In: Proceedings of 6th symposium on micro machine and human science, Nagoya, Japan, pp 39–43

Kim JJ, Park SY, Lee JJ (2009) Experience repository based particle swarm optimization for evolutionary robotics. In: ICCAS-SICE. IEEE, Washington, pp 2540–2544

Li XD, Engelbrecht AP (2007) Particle swarm optimization: an introduction and its recent developments. In: Geneticevolutionary computation conference, pp 3391–3414

Liang JJ, Qin AK, Suganthan PN, Baskar S (2006) Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Trans Evol Comput 10(3):281–295

Mahdavi M, Fesanghary M, Damangir E (2007) An improved harmony search algorithm for solving optimization problems. Appl Math Comput 188(2):1567–1579

Mann HB, Whitney DR (1947) On a test of whether one of two random variables is stochastically larger than the other. Ann Math Stat 18(1):50–60

Onwubolu GC, Babu BV (2004) New optimization techniques in engineering, vol 141. Springer, Berlin

Ragsdell KM, Phillips DT (1976) Optimal design of a class of welded structures using geometric programming. ASME J Eng Ind 98(3):1021–1025

Rahnamayan S, Tizhoosh HR, Salama MMA (2008) Opposition-based differential evolution. IEEE Trans Evol Comput 12(1):64–79

Ratnaweera A, Halgamuge SK, Watson HC (2004) Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. IEEE Trans Evol Comput 8(3):240–255

Sandgren E (1990) Nonlinear integer and discrete programming in mechanical design optimization. J Mech Des 112:223

Shi Y, Eberhart R (1998) A modified particle swarm optimizer. In: IEEE international conference on evolutionary computation proceedings. The 1998 IEEE world congress on computational intelligence. IEEE, Los Alamitos, pp 69–73

Shi Y, Eberhart Y (1998) Parameter selection in particle swarm optimization. In: Androutsos D, Plataniotis KN, Venetsanopoulos AN (eds) Evolutionary programming VII. Springer, Berlin, pp 591–600

Williamson DF, Parker RA, Kendrick JS (1989) The box plot: a simple visual method to interpret data. Ann Intern Med 110(11):916

Zhan ZH, Zhang J, Li Y, Chung HSH (2009) Adaptive particle swarm optimization. IEEE Trans Syst Man Cybern Part B 39(6):1362–1381

Zhang W, Li H, Zhang Z, Wang H (2008) The selection of acceleration factors for improving stability of particle swarm optimization. In: Fourth international conference on natural computation 2008 (ICNC'08), vol 1. IEEE, Haikou, pp 376–380