Opposition-Based Initialization and a Modified Pattern for Inertia Weight (IW) in PSO

Mehr Umer Farooq, Akhlaque Ahmad, and Abdul Hameed
Department of Computing and Technology, Iqra University Islamabad, Pakistan
Email: mehrumerfarooq@gmail.com, akhlaque.ahmad@iqraisb.edu.pk, hameed@iqraisb.edu.pk

***Abstract*—Particle Swarm Optimization (PSO) is an evolutionary computing algorithm and is successfully used to solve complex real world optimization problems. Due to the complex nature of optimization problems, PSO endures the problems like premature convergence or being trapped in local minima, to avoid such situation the role of swarm initialization is very important. In this research we propose a new method to initialize the swarm particles on the basis of Generalized Opposition-based Learning (GOBL). The aim for GOBL strategy is to have an initial swarm with already fittest particles to set a solid ground for the rest of PSO algorithm to execute. Moreover, a strategy for linearly decreasing Inertia Weight has been proposed to equalize the proportions of exploration as well as exploitation capabilities of particles during the search process. The motivation behind incorporating the changes in standard PSO is to evade the earlier convergence and to help the algorithm in escaping from being trapped in local minimum. To assess the performance of proposed PSO variant, we practiced this algorithm on 8 different benchmark functions and results were compared with 4 other PSO versions found in literature. From the results analysis it is apparent that projected changes in the PSO increases its overall performance and efficiency especially when dealing with the noisy optimization problems. Also the proposed algorithm performs better and is more robust as compared to other algorithms for achieving desired results.**

***Keywords*—Particle Swarm Optimization; Function Optimization; Inertia Weight; Generalized Opposition-based Learning.**

I. Introduction

SWARM intelligence is an aggregation of nature stirred searching methods based on population of individuals. Particle Swarm Optimization (PSO) algorithm is one of the stochastic population-based optimization methods that manifests the self-learning and shared comportment of flocking birds and fish schooling with an objective to find food [1], [2]. For the last about two decades, PSO, is being considered a very successful algorithm for solving *complex*, *multi-dimensional optimization problems* [3] and *complex real-world problems* [4].

Since the very first introduction of PSO, several variations have been introduced into it with primary intention to improve its overall working. The authors in [5] introduced a binary version of PSO, where decision making capability of all particles is specified in the form of 1 *or* 0. The idea of introducing inertia weight in PSO is developed in [6], [7]. The inertia weight is practiced to limit the exposition of the earlier velocity into present velocity and is represented in velocity update equation (1) and (2).

$$V_{id}^{t+1} = w * V_{id}^{t} + c_1 r_{1id}(P_{id} - X_{id}^{t}) + c_2 r_{2id}(P_{gd} - X_{id}^{t}) \quad (1)$$

Where

$$X_{id}^{t+1} = X_{id}^{t} + V_{id}^{t+1}, i = 1, \ldots, N; d = 1, \ldots, D \quad (2)$$

Here t denotes iteration number. V_{id}^{t} denotes the velocity and X_{id}^{t} denotes the position of the particle i. w is called the inertia weight [8], c_1 and c_2 are known as the acceleration coefficients [3]. However, r_{1id} and r_{2id} are vectors, which are arbitrarily produced from the range $[0, 1]$, P_{id} also known as personal best, is the best result of the i_{th} particle established so far and P_{gd} is identified as global best, is the best result found so far by the collaborated effort of all particles. The $c_1 r_{1id}(P_{id} - X_{id}^{t})$ part of the equation (1) is known as cognitive constituent whereas $c_2 r_{2id}(P_{gd} - X_{id}^{t})$ part is known as social constituent [8].

In equation (1), the value of inertia weight w remains fix (constant) throughout the execution of the algorithm. However, instead of having a constant value of inertia weight the authors in [9] proposed *PSO-RANDIW (Randomly Decreasing Inertia Weight)* and is elaborated in equation (3).

$$w = 0.5 + \frac{1}{2} \times rand() \quad (3)$$

It is evident from literature that, all algorithms where population of individuals is involved, generally undergo early convergence because of the extremely composite nature of the optimization problem [4]. Being a population-based algorithm, PSO also exhibits the problem of premature convergence. Hence, a plethora of research has been directed to overcome these limitations [10], [11].

In optimization problems, the role of swarm initialization is vital and the chance of exploiting the ineffective areas of the search domain increases with the random initialization of the swarm population [12]. Hence, the need for intelligent swarm initialization methods based on some realistic approaches is essential to get the better search results.

The key input in this research is the integration of GOBL in initializing the swarm. We propose a modified pattern for inertia weight in the search process for two reasons. One is to balance the exploration and exploitation whereas the other one is to overcome the problem of trapping in local optima by the particles.

978-1-5090-5795-5/17/$31.00

II. Background and Related Work

The authors in [13] introduced the concept of OBL in machine intelligence, where the author deliberated the key opinion by incorporating counter-estimates, contrary numbers, anti-chromosomes, counter-actions and contrary weights in machine erudition algorithms. Incorporation of OBL technique in PSO undoubtedly improved its performance [14]. Different applications of OBL are presented in [13], [15].

In [16] a novel algorithm called Opposition-based PSO (*OPSO*) by incorporating OBL for initial swarm generation, jumping of generation and improving the swarm particle with global highest degree measure was proposed. The aim of *OPSO* was to heighten the efficiency of PSO and to handle the noisy optimization problems effectively.

The authors in [17] proposed a *OPSO* technique, employing opposition-based initialization, jumping of generation with changing search space area, and using dynamic Cauchy mutation operator for each iteration was used for the global better position for the intent of spreading the lookup region for the better particle.

The approach in [18] introduced a novel Opposition-based Comprehensive Learning PSO also known as (*OCLPSO*), where comprehensive learning was inspired by opposition-based exemplars. The process was stretched over three different stages i.e. (1) *Opposition-based Population Initialization* (2) *Opposition-based Exemplar Selection* and (3) *Ceased Particle Jumping*.

Generalized Opposition-based Learning PSO (*GOPSO*) was introduced in [19]. The aim of algorithm was to cope with the problem of premature convergence in PSO and was accomplished through incorporating the concept of *GOBL* and Cauchy mutation in PSO. The incorporation of GOBL helped in faster convergence whereas, Cauchy mutation assisted in escaping the particles, which were trapped in local optimum. The results of *GOPSO* were promising in many optimization problems. The authors also presented generalized opposition-based PSO with dynamic Population size (*DP-GOPSO*) and an extension of *GOPSO* algorithm where dynamic population was included.

The authors in [12] introduced O-PSO by integrating the OBL technique for swarm initialization. In this proposed method, the swarm was generated by combing the best particles from both randomly initialized swarm and opposite of that swarm. The execution of suggested O-PSO was collated with traditional PSO by testing it on different benchmark functions and the results were promising.

A multi-start opposition-based PSO algorithm with an adaptive velocity was proposed in [20]. Three basic stages were involved in this proposed method. Detail of the stages are enumerated below:-

1) First one is called the initialization stage, where the authors used OBL approach with an objective to increase the searching capability.
2) In the second stage, a different version of the adaptive velocity *(established on the differential operator)* was applied for the purpose to increase the optimization capability of the swarm particles.
3) Finally a re-initialization technique based on two diversity measures for the swarm was employed to vanquish the issue of earlier convergence and stagnation.

The authors in [21] introduced probabilistic opposition-based PSO with velocity clamping and inertia weights and named the same as *OvcPSO*. In *OvcPSO* the prime areas under research were OBL, velocity clamping and inertia weights.

In [22] a variant of PSO called *OpbestPSO* was introduced by employing the methodology of OBL. In *OpbestPSO* algorithm, initially the swarm was randomly initialized and their opposites were calculated. From the two swarms (i.e. randomly initialed swarm and its opposite swarm) the fittest particles were retained as swarm for the rest of algorithm. Secondly, to improve the performing of PSO, for each iteration, the authors used the technique of OBL on $pbest$ particle of the swarm. For this purpose, at each iteration, authors calculated the opposite of $Xpbest$ i.e. Opp_Xpbest using equation (4). The fitness of both $Xpbest$ particle and Opp_Xpbest was evaluated, and only the better one was retained.

$$opp\_xpbest = k.(A + B) - xpbest \tag{4}$$

For each iteration the value of k is randomly generated between $[0, 1]$. However, in each iteration the value of k remains same for every swarm particle. A and B are the search boundaries. If the particle Opp_xpbest escapes the boundary $[X_{min}, X_{max}]$, then using equation (5) it is re-initialized.

$$opp\_xpbest(j) = A(j) + (B(j) - A(j)).rand(0, 1) \tag{5}$$

In this paper we have proposed a new technique for swarm initialization. Moreover, to avoid premature convergence in PSO a slightly different pattern for inertia weight is also presented in Section III.

III. Proposed Methodology

A. Opposition-based Learning (OBL)

The theme behind the OBL is keeping in view the approximation and opponent of the approximation simultaneously for the purpose of achieving a good approximation for underlying candidate solution. This methodology is very beneficial, especially when dealing with the noisy optimization problems, where the optimal solution might have been displaced due to noise. Actually, considering the estimate of opposite (*viewing the opposite aspect*) proposes an other opportunity to judge the displacement of optimum solution [16]. The same idea can be incorporated to a broad variety of optimization problems. In this research, the idea of OBL has been incorporated with PSO, however, due to its generalization the same can also be practiced with other swarm based algorithms [16].

As per the social manifestation for estimable and bad, if one individual is good then, his contrary is more likely to be a bad. Moreover, at the same time, it is less likely that two opposites are entirely estimable and entirely inestimable simultaneously. We utilize this natural lineament of mankind and present a

similar technique for initializing the swarm particles in PSO [12].To pursue the OBL technique further, it will be helpful if we first explain the opposite numbers [13].

Definition 1: Let x be a real number i.e. R in an interval $[a, b]$ $(x \ \epsilon \ [a, b])$; the opponent of this number i.e. $\breve{x}$ is delineated as:-

$$\breve{x} = a + b - x \tag{6}$$

Likewise, similar description can be stretched for multiple dimensions as mentioned below [15].

Definition 2: Let $P(x_1, x_2, \ldots, x_n)$ is a point in n-dimensional region, where $x_1, x_2, \ldots, x_n \ \epsilon \ R$ and $x_i \ \epsilon \ [a_i, b_i] \ \forall_i \ \epsilon \ \{1, 2, \ldots, n\}$. Then the opposite of this point P can be delineated as $\breve{P}$ = ($\breve{x}_1$,$\breve{x}_2$,...,$\breve{x}_n$) where:

$$\breve{x}_i = a_i + b_i - x_i \tag{7}$$

And by employing the definitions of opposite point, we can describe the particle of the swarm. A single particle in the swarm P_i can be delineated as:-

$P_i \ \epsilon \ [a, b]$, where $i = 1, 2, 3 \ldots D$, and $a, b \ \epsilon \ R$. D stands for dimensions of the optimizing problem and R denotes real numbers. Apiece particle P_i has a singular opponent P_{Opi} in predetermined n-dimensions and can be delineated as under:-

$$P_{Opi} = a + b - P_i \tag{8}$$

Where $i = 1, 2, 3 \ldots D$, and $a, b \ \epsilon \ R$, D stands for dimensions of the optimizing problem and R denotes real numbers.

B. Threshold Value

From the dynamic information collected from the current population, a threshold measure / rate of the fitness function is calculated using equation (9) introduced by [3].

$$U = K \times \frac{\sum_{i=1}^{N} pop\_value_i}{N} \tag{9}$$

pop_value_i is known as the fitness value of i^{th} particle. N represents the swarm size. The value of K is limited to 0.2, which is obtained through trial and error. In the present techniques of error and trail, the authors have practiced various extensive numerical tests / examinations on the basis of case studies to determine the value of K as mentioned in [3].

C. Velocity Clamping

Velocity clamping was introduced by Eberhart and Kennedy to help the particles remain within the limits of searching domain and to take good step size for combing the search domain. With the absence of velocity clamping, the optimization procedure will become prone to explode and the particles will change their positions very rapidly [23]. With large value of V_{max} the particles may move unpredictably and chances are present that they may even jump over the optimal solution. Conversely, if the value of V_{max} is quite small then the movement of particles may be very restricted, the swarm may not be efficient to examine the promising regions of lookup space. Hence, initializing V_{max} is very important and is typically chosen from 0.1 to 1.0 times the lookup range of the potential solution space [24], [25].

To restrict the swarm particles to remain within the lookup area, the velocity in d^{th} dimension is constrained between $[-V_{dmax} \ \ V_{dmax}]$ using equation (10) and (11).

$$v_{id} = -V_{dmax} \ \ if \ \ v_{id} < -V_{dmax} \tag{10}$$

$$v_{id} = V_{dmax} \ \ if \ \ v_{id} > V_{dmax} \tag{11}$$

D. Proposed Method

Swarm of size N, and velocities of all the swarm particles are randomly initialized. Fitness measures for all the particles in the swarm are calculated using the objective function. Based on the acquired fitness measures of all the particles, a threshold value is ascertained using equation (9). Threshold value is applied to split up the initial swarm into two separate swarms namely Best and the Worst. For the Worst swarm, we calculated the opposite of all the particles using equation (8). To get our final swarm of size N, we combine the Best swarm with the lately calculated opposite of the Worst swarm. Also we calculated the fitness of all the particles for the purpose to identify the $pbest$ and $gbest$ particles within the swarm, which are used through the rest of the procedure. After that, the main body of PSO starts, which lasts till the termination criteria is finally met. As soon as the loop of PSO starts, we calculated the opposite of $pbest$ i.e. $Opbest$. Once we have calculated the $Opbest$, we compared it with the $pbest$. If the fitness rate of $Opbest$ is higher than the fitness rate of $pbest$, we replaced $Opbest$ with $pbest$, otherwise $pbest$ was unchanged. Velocity and locality of all the particles present in the swarm is rationalized using equation (1) and (2) respectively.

Finally, $pbest$ and $gbest$ particles are updated and the loop continues till the stopping criteria is satisfied. After the termination of loop we get our particle with the optimum value.

E. Proposed Inertia Weight (IW) Pattern

The LDIW strategy [26] heighten the efficiency and overall functioning of PSO algorithm. From empirical results, it is proved that value of Inertia Weight is chosen between 0.9 and 0.4 to get the admirable results [26]. However, the shortcoming associated with LDIW technique is that the PSO can easily struck in local optima especially when the optimization function has multiple optimum solutions [26]. To curb the shortcoming, we have proposed a strategy in which inertia weight is linearly lessen from 0.9 till 0.4 for the first half of the iterations. And then repeat the same strategy for remaining half of the iterations. Same strategy is shown in Figure (1).

With the integration of aforementioned strategy, at two points during the execution, the value of inertia weight is high i.e. at the beginning of the algorithm and when half of the iterations are done. Due to the elevated value of inertia weight, the particles focus on scrutinizing the lookup area. Similarly, at two points during the execution, the value of inertia weight is approaching its minimum value, firstly when the iterations are approaching its half and secondly near the

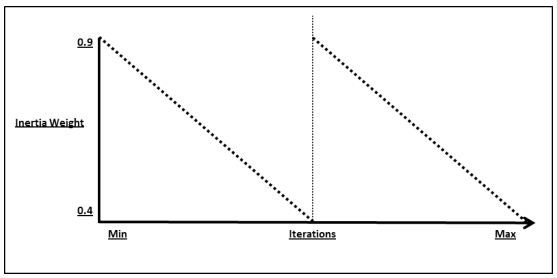


Fig. 1. Proposed Inertia Weight Strategy

end of iterations. Due to the minimum value of inertia weight the focus of the particles is to locally exploit the search region under consideration to obtain a better solution.

F. Algorithm of Proposed Technique

Algorithm and necessary steps involved in the proposed technique are illustrated in Algorithm (1).

Initialize the swarm X randomly, of population size N;
Initialize the velocity V;
Evaluate the fitness of all particles i.e. X;
Evaluate the Threshold Value using equation (9);
Divide the swarm X into two sets using Threshold value i.e. ***Best Particles*** ***(X_{Best})*** and the ***Worst Particles*** ***(X_{Worst})***;
Calculate opposite of Worst Particles i.e. OX_{Worst} using equation (8);
Merge / Combine particles from the set $\{X_{Best}$, $OX_{Worst}\}$ i.e. X;
Evaluate the fitness of all particles i.e. X;
Calculate $pbest$ and $gbest$;
while *termination criteria i.e. Maximum number of FEs* **do**
  for i = 1 *to* N **do**
    Calculate opposite of $pbest$ i.e. $Opbest$;
    Select the fittest one from set $\{pbest, Opbest\}$;
    Update velocity and position components of all the particles using equation (1) and (2) respectively;
    Update $pbest$ and $gbest$;
  end
end

Algorithm 1: Algorithm of Proposed Technique

IV. Experimental Framework

A. Benchmark Problems

For the purpose of comparison with other variants of PSO and to examine the working capacity of proposed algorithm, in the present research, we have used a suite of various well known test functions. The suite contains eight benchmark functions for minimization. Functions from f_1 to f_3 and f_8 are uni-modal. However, functions from f_4 through f_6 are multi-modal and lastly f_7 is a quartic noisy function. Delineation of the test functions is presented in Table I.

TABLE I
Basic Information Regarding Test Functions

Function Name	Solution Space	f_{min}
$f_1(x) = \sum_{i=1}^{n} x_i^2$	$[-100, 100]$	0
$f_2(x) = \sum_{i=1}^{n} [\sum_{j=1}^{i} x_j]^2$	$[-100, 100]$	0
$f_3(x) = \sum_{i=1}^{n-1} 100.(x_{i+1} - x_i^2)^2$ $+(1 - x_i)^2$	$[-100, 100]$	0
$f_4(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}}) + 1$	$[-600, 600]$	0
$f_5(x) = -20 * exp(-0.2 * \sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2})$ $-exp(\frac{1}{n} \sum_{i=1}^{n} cos(2.\pi.x_i)) + 20 + e$	$[-32, 32]$	0
$f_6(x) = 10.n + \sum_{i=1}^{n} (x_i^2 - 10.cos(2.\pi.x_i))$	$[-5.12, 5.12]$	0
$f_7(x) = \sum_{i=1}^{n} ix_i^4 + random(0, 1)$	$[-1.28, 1.28]$	0
$f_8(x) = \sum_{i=1}^{n} \lvert x_i \rvert - \prod_{i=1}^{n} \lvert x_i \rvert$	$[-10, 10]$	0

Test functions included in the above mentioned suite are specially designed with the intention to test the various properties of optimization algorithms and are also used in [16], [17], [22] to test the properties of modified PSO algorithms.

B. Parameters for Proposed PSO

For experiments, a swarm of 30 particles is used. Dimension of the problem is chosen 30. Maximum number of Function Evaluations (FEs) for a function is set as 1e+5. Values for personal and social cognizance factors i.e. c_1 as well as c_2 are limit to 2.05. Maximum and minimum value of inertia weight is 0.9 and 0.4 respectively. Finally, mean is calculated over 30 runs per function.

C. Termination / Stopping Criteria

Execution of the algorithm is terminated either by reaching a maximum predefined limit of FEs or whenever a best-run-error (*E*) is achieved. *E* is defined as an absolute difference between the global optimum value and the best solution value obtained in a single run. E is calculated using equation (12).

$$E = |f(x) - f(x^*)| \leq e \tag{12}$$

In equation (12), $f(x)$ is the best solution achieved during the current run, whereas, $f(x^*)$ is the global optimum value. *e* is known as the threshold error value. In the present research, the value of *e* has been taken as $1e-08$ for all the functions.

V. Results Analysis

The formulated technique is practiced on 8 eminent benchmark unconstrained optimization functions for 30 dimensions. Once the population has been initialized, it is kept same for a single run for the purpose that the alterations in functioning highlight the algorithms own behavior to make a reasonable assessment among the algorithms. In this paper mean and standard deviation are used as preferred metrics for measuring the quality of solutions. Mean and Standard Deviation of solution values obtained from 30 independent runs are tabulated in Table II. Although the maximum limit of iterations or FEs per run is $1e+5$, however, convergence graphs of 8 optimization functions for only the initial 2000 FEs are illustrated in Table

TABLE II
MEAN AND STANDARD DEVIATION OF SOLUTION VALUES OVER 30 INDEPENDENT RUNS

f(x)	**PSO-w** [6]		**OPSO** [16]		**OPSO** [17]		**OpbestPSO** [22]		**Proposed PSO**	
	Mean	**Std. Dev.**	**Mean**	**Std. Dev.**	**Mean**	**Std. Dev.**	**Mean**	**Std. Dev.**	**Mean**	**Std. Dev.**
f_1	6.144E+00	1.021E+01	4.000E-04	2.100E-03	7.095E+02	1.375E+03	1.951E-01	8.972E-01	9.840E-09	2.380E-10
f_2	3.328E-07	2.623E-07	1.484E-06	2.950E-06	1.679E-05	3.805E-05	3.704E-07	3.080E-07	9.020E-06	8.950E-06
f_3	1.100E+04	2.826E+04	2.716E+01	4.314E-01	7.442E+05	1.155E+06	2.987E+01	5.590E+00	3.830E-03	6.920E-03
f_4	5.949E-01	5.681E-01	2.000E-03	1.140E-02	2.839E+00	8.995E+00	4.180E-02	1.265E-01	1.020E-07	2.060E-07
f_5	2.818E+00	3.775E+00	7.220E-02	3.955E-01	5.939E+00	4.025E+00	2.521E-02	8.870E-02	2.290E-04	8.740E-05
f_6	5.828E+01	3.116E+01	1.798E-01	9.851E-01	2.259E+02	1.119E+01	7.000E-04	1.500E-03	1.090E+00	3.070E+00
f_7	4.160E-02	7.500E-02	6.600E-03	5.600E-03	9.117E-01	4.908E-01	1.000E-04	7.240E-05	1.880E-05	1.070E-05
f_8	8.212E+00	9.131E+00	6.000E-04	3.600E-03	1.165E+01	1.499E+01	2.720E-02	3.270E-02	3.150E-06	1.210E-05

V. Success Rate (SR) is obtained using the equation (13). SR calculated for a total number of 30 runs is presented in Table III.

$$SR = \frac{Numbers\ of\ Total\ Thresholds\ Errors\ Achieved}{Total\ Number\ of\ Runs} \tag{13}$$

Results of performance comparison with competitor algorithms are presented in Table IV, where a single cell of table indicates that whether the performance of proposed algorithm is better than, less than or comparable with this algorithm. The value "+" indicates that the performance of proposed algorithm is better than this algorithm for the particular optimization function. The value "−" indicates that the performance of proposed algorithm is relatively less than the algorithm. Moreover, the value "≈" indicates that the performance of both algorithms is comparable.

A. Empirical Performance

From Table II it is apparent that proposed algorithm is more robust when compared to other contending algorithms for accomplishing craved results. The proposed algorithm has performed significantly better over PSO-w [6], OPSO [16], OPSO [17] and OpbestPSO [22] for functions $f_1, f_3 - f_5, f_7 - f_8$. For function f_2, the results are better over OPSO [17] and are nearly equal with OPSO [16]. However, PSO-w [6] and OpbestPSO [22] performed better over proposed algorithm. For function f_6, the results are better as compared with PSO-w [6] and OPSO [17]. However, the performance of OPSO [16] and Opbest [22] remained better over proposed algorithm.

Considering the experimental results, it can be concluded that the proposed methodology significantly improve the performance of PSO as compared to other PSO variants, which integrate OBL.

VI. CONCLUSION

From literature it is apparent that for optimization problems, the role and importance of swarm initialization cannot be ruled out. Therefore, in this research work, GOBL is integrated to initialize the PSO swarm and a modified inertia weight pattern has been proposed. The motivation is to have an initial swarm with already best particles. That will set a solid foundation for the rest of the algorithm to execute. The purpose behind using the proposed methodology for inertia weight is due to two reasons. One is to balance the exploration and exploitation whereas the other one is to subjugate the trouble of trapping in local optima.

TABLE III
SUCCESS RATE (%) OVER 30 RUNS

f(x)	**PSO-w** [6]	**OPSO** [16]	**OPSO** [17]	**OpbestPSO** [22]	**Proposed PSO**
f_1	36.67	96.67	0.00	6.67	100
f_2	100.00	63.33	53.33	100.00	0.00
f_3	0.00	0.00	0.00	0.00	0.00
f_4	40.00	96.67	0.00	10.00	13.33
f_5	20.00	96.33	0.00	0.00	0.00
f_6	0.00	96.67	0.00	6.67	6.67
f_7	0.00	0.00	0.00	0.00	0.00
f_8	6.67	93.33	0.00	0.00	30.0

To evaluate the performance of our PSO variant, we applied the same on 8 benchmark functions. Moreover, we compared the results with 4 other PSO versions found in literature, to judge the performance of our proposed algorithm. From the solutions presented in Table II, it is evident that projected changes increase the overall performance and efficiency of PSO especially when the optimization problems are noisy.

VII. FUTURE WORK

In this research we have shown that, our proposed methodology enables the PSO to find robust optima. Proposed methodology can be coupled with other variants of PSO for better performance.

However, as the convergence haste of the projected algorithm is not up to the mark, so it is suggested that, efforts should be directed towards introducing new parameters to improve its convergence speed. By doing so, computational complexity will be reduced and it will ensure earlier convergence and will avoid getting trapped in local minima.

REFERENCES

[1] Russ C Eberhart, James Kennedy, et al. A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micro machine and human science*, volume 1, pages 39–43. New York, NY, 1995.

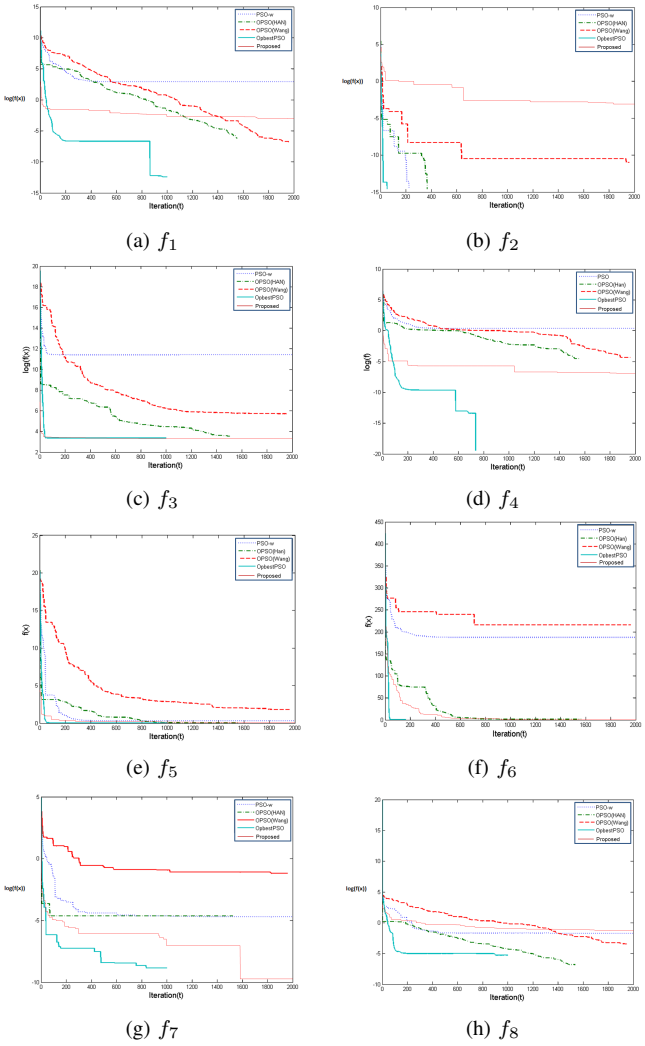
[2] James Kenndy and RC Eberhart. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948, 1995.

[3] Shafi Ullah Khan, Shiyou Yang, Luyu Wang, and Lei Liu. A modified particle swarm optimization algorithm for global optimizations of inverse problems. *IEEE Transactions on Magnetics*, 52(3):1–4, 2016.

TABLE IV
COMPARISON OF RESULTS

$f(x)$	**PSO-w** [6]	**OPSO** [16]	**OPSO** [17]	**OpbestPSO** [22]
f_1	+	+	+	+
f_2	−	≈	+	−
f_3	+	+	+	+
f_4	+	+	+	+
f_5	+	+	+	+
f_6	+	−	+	−
f_7	+	+	+	+
f_8	+	+	+	+

TABLE V
CONVERGENCE GRAPHS FOR FUNCTIONS $f_1 - f_8$



(a) f_1 (b) f_2

(c) f_3 (d) f_4

(e) f_5 (f) f_6

(g) f_7 (h) f_8

[4] MR Tanweer, S Suresh, and N Sundararajan. Mentoring based particle swarm optimization algorithm for faster convergence. In *Evolutionary Computation (CEC), 2015 IEEE Congress on*, pages 196–203. IEEE, 2015.

[5] James Kennedy and Russell C Eberhart. A discrete binary version of the particle swarm algorithm. In *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, volume 5, pages 4104–4108. IEEE, 1997.

[6] Yuhui Shi and Russell Eberhart. A modified particle swarm optimizer. In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, pages 69–73. IEEE, 1998.

[7] A Nikabadi and M Ebadzadeh. Particle swarm optimization algorithms with adaptive inertia weight: A survey of the state of the art and a novel method. *IEEE journal of evolutionary computation*, 2008.

[8] Ran Cheng and Yaochu Jin. A competitive swarm optimizer for large scale optimization. *Cybernetics, IEEE Transactions on*, 45(2):191–204, 2015.

[9] Russell C Eberhart and Yuhui Shi. Tracking and optimizing dynamic systems with particle swarms. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, pages 94–100. IEEE, 2001.

[10] Xiang Yu and Xueqing Zhang. Enhanced comprehensive learning particle swarm optimization. *Applied Mathematics and Computation*, 242:265–276, 2014.

[11] Kezong Tang, Zuoyong Li, Limin Luo, and Bingxiang Liu. Multi-strategy adaptive particle swarm optimization for numerical optimization. *Engineering Applications of Artificial Intelligence*, 37:9–19, 2015.

[12] Hajira Jabeen, Zunera Jalil, and Abdul Rauf Baig. Opposition based initialization in particle swarm optimization (o-pso). In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, pages 2047–2052. ACM, 2009.

[13] Hamid R Tizhoosh. Opposition-based learning: a new scheme for machine intelligence. In *null*, pages 695–701. IEEE, 2005.

[14] Biplab Mandal and Tapas Si. Opposition based particle swarm optimization with exploration and exploitation through gbest. In *Advances in Computing, Communications and Informatics (ICACCI), 2015 International Conference on*, pages 245–250. IEEE, 2015.

[15] Hamid R Tizhoosh. Reinforcement learning based on actions and opposite actions. In *International conference on artificial intelligence and machine learning*, volume 414, 2005.

[16] Lin Han and Xingshi He. A novel opposition-based particle swarm optimization for noisy problems. In *Natural Computation, 2007. ICNC 2007. Third International Conference on*, volume 3, pages 624–629. IEEE, 2007.

[17] Hui Wang, Hui Li, Yong Liu, Hui Li, and Sanyou Zeng. Opposition-based particle swarm algorithm with cauchy mutation. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 4750–4756. IEEE, 2007.

[18] Zhangjun Wu, Zhiwei Ni, Chang Zhang, and Lichuan Gu. Opposition based comprehensive learning particle swarm optimization. In *Intelligent System and Knowledge Engineering, 2008. ISKE 2008. 3rd International Conference on*, volume 1, pages 1013–1019. IEEE, 2008.

[19] Hui Wang, Zhijian Wu, Shahryar Rahnamayan, Yong Liu, and Mario Ventresca. Enhancing particle swarm optimization using generalized opposition-based learning. *Information Sciences*, 181(20):4699–4714, 2011.

[20] Massimiliano K. A multi-start opposition-based particle swarm optimization algorithm with adaptive velocity for bound constrained global optimization. *Journal of Global Optimization*, 55(1):165–188, 2013.

[21] Farrukh Shahzad, Sohail Masood, and Naveed Kazim Khan. Probabilistic opposition-based particle swarm optimization with velocity clamping. *Knowledge and information systems*, 39(3):703–737, 2014.

[22] Tapas Si, Arunava De, and A. K. Bhattacharjee. Particle swarm optimization with generalized opposition based learning in particle's pbest position. In *Circuit, Power and Computing Technologies (ICCPCT), 2014 International Conference on*, pages 1662–1667. IEEE, 2014.

[23] Daniel Bratton and James Kennedy. Defining a standard for particle swarm optimization. In *2007 IEEE swarm intelligence symposium*, pages 120–127. IEEE, 2007.

[24] Alec Banks, Jonathan Vincent, and Chukwudi Anyakoha. A review of particle swarm optimization. part i: background and development. *Natural Computing*, 6(4):467–484, 2007.

[25] Mengqi Hu, Tsai-Fu Wu, and Jeffery D Weir. An adaptive particle swarm optimization with multiple adaptive methods. *Evolutionary Computation, IEEE Transactions on*, 17(5):705–720, 2013.

[26] JC Bansal, PK Singh, Mukesh Saraswat, Abhishek Verma, Shimpi Singh Jadon, and Ajith Abraham. Inertia weight strategies in particle swarm optimization. In *Nature and Biologically Inspired Computing (NaBIC), 2011 Third World Congress on*, pages 633–640. IEEE, 2011.