*I.J. Modern Education and Computer Science*, **2017, 8, 49-56**
Published Online August 2017 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2017.08.07

# A Modified Particle Swarm Optimization Algorithm based on Self-Adaptive Acceleration Constants

**Sudip Mandal**
Department of Electronics and Communication Engineering,
Global Institute of Management and Technology, Krishna Nagar, India-741102.
Email: sudip.mandal007@gmail.com

Received: 14 March 2016; Accepted: 25 July 2017; Published: 08 August 2017

***Abstract*—**Particle Swarm Optimization (PSO) is one of most widely used metaheuristics which is based on collective movement of swarm like birds or fishes. The inertia weight (*w*) of PSO is normally used for maintaining balance between exploration and exploitation capability. Many strategies for updating the inertia weight during iteration were already proposed by several researchers. In this paper, a Modified Particle Swarm Optimization (MPSO) algorithm based on self-adaptive acceleration constants along with Linear Decreasing Inertia Weight (LDIW) technique is proposed. Here, in spite of using fixed values of acceleration constants, the values are updated themselves during iteration depending on local and global best fitness value respectively. Six different benchmark functions and three others inertia weight strategies were used for validation and comparison with this proposed model. It was observed that proposed MPSO algorithm performed better than others three strategies for most of functions in term of accuracy and convergence although its execution time was larger than others techniques.

***Index Terms*—**Metaheuristic; Optimization; Modified Particle Swarm Optimization (MPSO); Inertia Weight; Acceleration Constant.

## I. Introduction

Metaheuristic optimization techniques have become very popular over the last two decades [21]. They have been mostly inspired by very simple concepts. The inspirations are typically related to physical phenomena, animals' behaviors, or evolutionary concepts.

Regardless of the differences between the meta-heuristics, a common feature is the division of the search process into two phases: exploration and exploitation. The exploration phase refers to the process of investigating the promising area(s) of the search space as broadly as possible. An algorithm needs to have stochastic operators to randomly and globally search the search space in order to support this phase. However, exploitation refers to the local search capability around the promising regions obtained in them exploration phase. Finding a proper balance between these two phases is considered a challenging task due to the stochastic nature of meta-heuristics.

There is a question here as to why metaheuristics have become remarkably common. The answer to this question can be summarized into four main reasons: simplicity, flexibility, derivation- free mechanism, and local optima avoidance.

First, metaheuristics are fairly simple. The simplicity allows computer scientists to simulate different natural concepts, propose new meta-heuristics, hybridize two or more metaheuristics, or improve the current meta-heuristics.

Second, flexibility refers to the applicability of metaheuristics to different problems without any special changes in the structure of the algorithm assuming problems as black boxes. So, all a designer needs is to know how to represent his/her problem for meta-heuristics.

Third, the majority of meta-heuristics have derivation-free mechanisms i.e. meta-heuristics optimize problems stochastically. The optimization process starts with random solution(s), and there is no need to calculate the derivative of search spaces to find the optimum. This makes meta-heuristics highly suitable for real problems with expensive or unknown derivative information.

Finally, meta-heuristics have superior abilities to avoid local optima compared to conventional optimization techniques. This is due to the stochastic nature of meta-heuristics which allow them to avoid stagnation in local solutions and search the entire search space extensively.

One of the interesting branches of the population-based metaheuristics is Swarm Intelligence (SI). The inspirations of SI techniques originate mostly from natural colonies, flock, herds, and schools. Here, the search agents navigate using the simulated collective and social intelligence of creatures. Some of the most popular SI techniques are Particle Swarm Optimization (PSO) [1], Ant Colony Optimization (ACO) [22], Artificial Bee Colony (ABC) [23] and Bat Algorithm (BA) [24] etc.

Among these, Particle Swarm Optimization (PSO) is one of the most efficient optimization strategies for continuous nonlinear optimization problems. J. Kennedy *et al.* [1] first proposed PSO that is inspired by the collective social behaviors of swarm like movements of flocks of birds or schools of fish in search of foods. The main advantages of PSO are that PSO is very simple and efficient in nature (no need of gradient of the problem etc.); there are few parameters (no need of different complex genetic operators etc.) needed to be adjusted. In addition to the huge number of theoretical works, PSO has been applied in various fields of study DC-Side Voltage Control [25], Task Scheduling [26], Workflow Scheduling [27] and Transportation Network design [28] problems etc.

In case of PSO, a particle (i.e. bird or fish) denotes a potential solution for the optimization problem. A set of particles is known as a swarm, where particles are initially distributed or positioned in random manner in *d*-dimensional search space of the problem. Swarm is flown through the search space and the position of each particle is updated based on the experiences (fitness value at that point) of all neighbors particle including itself [2]. Every particle is considered as intelligent and knows its own current fitness value, its own best value so far (locally best solution), the best fitness value of the whole swarm (globally best solution), and its own velocity [3].

For, *d*-dimensional optimization problem, the position of *i*-th particle of a swarm (consist of $N$ particles) at $t$-th iteration is given as $X_{i,d}^{t} = (x_{i1}, x_{i2}, \ldots., x_{id})$ and the velocity is represented by $V_{i,d}^{t} = (v_{i1}, v_{i2}, \ldots., v_{id})$. Locally best solution by *i*-th particle at current iteration is given as $P_{best,i,d}^{t} = (P_{i1}, P_{i2}, \ldots., P_{id})$ and global best solution is denoted by $G_{best,d}^{t} = (G_1, G_2, \ldots., G_d)$. As iteration proceeds, the velocity and position of the particles are updated according to following rules [1].

$$V_{i,d}^{t+1} = V_{i,d}^{t} + C_1 * rand(1,d) \odot \left(P_{best,i,d}^{t} - X_{i,d}^{t}\right) + C_2 * rand(1,d) \odot \left(G_{best,d}^{t} - X_{i,d}^{t}\right) \quad (1)$$

$$X_{i,d}^{t+1} = V_{i,d}^{t+1} + X_{i,d}^{t} \quad (2)$$

Where $C_1$ and $C_2$ are called as acceleration constants, also named as cognitive learning rate and social learning rate respectively. $rand(1,d)$ is generate a *d*-dimensional array of random values within [0,1]. $\odot$ denotes element wise multiplication.

However, one of the main reasons behind the success of a metaheuristic is a delicate balance between exploration and exploitation capability of the algorithm. Several authors [4-18] proposed different methods to achieve better accuracy and convergence. However, in this paper, we have proposed a Modified Particle Swarm Optimization (MPSO) Algorithm based on self-adaptive acceleration constants. The rest of this paper is organized as follows. Section II describes different existing PSO strategies. The proposed Modified PSO is described Section III which is followed by results and analysis section. Next, conclusion and references are provided.

## II. Related Works

To get a better control between the global and local search characteristics of PSO, Shi and Eberhart [4] proposed a modified PSO where the velocity of each particle is updated based on inertia weight ($\omega$). So, the velocity update rule is modified according to following equation.

$$V_{i,d}^{t+1} = V_{i,d}^{t} * \omega^t + C_1 * rand(1,d) \odot \left(P_{best,i,d}^{t} - X_{i,d}^{t}\right) + C_2 * rand(1,d) \odot \left(G_{best,d}^{t} - X_{i,d}^{t}\right) \quad (3)$$

Recently, researchers give lots of attentions to the inertia weight parameter for improving the performance of original PSO. Lots of strategies were already proposed for updating inertia weight during the course of iteration.

In 1998, Shi and Eberhart [4] proposed Constant Inertia Weight (CIW) technique where they claimed that large constant value of Inertia Weight is suitable for exploration while a small constant value of Inertia Weight is suitable a exploitation. So, CIW can be described using following equation.

$$\omega^t = Constant \quad (4)$$

Further, in case of Random Inertia Weight (RIW) [5], the value of inertia weight is selected in random manner and it is very efficient to find out the optima in a dynamic system. For RIW, the value of inertia weight is assigned using following equation

$$\omega^t = 0.5 + rand/2 \quad (5)$$

Where $rand$ is a function that generates random number within [0, 1]. Therefore, value of inertia weight is uniformly distributed over [0.5, 1] and this technique partially solve the problem of selection for constant of CIW.

Linear Decreasing Inertia Weight (LDIW) [6-8] is very popular and efficient technique in improving the fine-tuning characteristics of the PSO where the value of inertia weight is linearly depend on the iteration number. In case of LDIW, the value of $\omega$ is linearly decreased from an initial large value ($\omega_{max}$) to a final small value ($\omega_{min}$) according to the following equation:

$$\omega^t = \omega_{max} - \left\{\frac{\omega_{max} - \omega_{min}}{t_{max}}\right\} \times t \quad (6)$$

Where $t$ is iteration index and $t_{max}$ denotes maximum number of iteration. LDIW has better efficiency over the others technique due to smooth transition from initial global search to local search during iteration process [20].

There are lots of others strategies for variation of inertia weight like Adaptive Inertia Weight [9], Sigmoid Increasing Inertia Weight [10], Chaotic Inertia Weight [11], Oscillating Inertia Weight [12], Global-Local Best Inertia Weight [13], Simulated Annealing Inertia Weight [14], Exponent Decreasing Inertia [15], Natural Exponent Inertia Weight Strategy [16], Logarithm Decreasing Inertia Weight [17], Nonlinear Decreasing Variant of

Inertia Weight [18] and Bat-PSO hybridization [19] etc. However, in this context, our proposed technique is compared with only other 3 strategies which CIWPSO, RIWPSO and LDIWPSO for validation purpose

## III. Methodolody

Fine parametric tuning of evolutionary algorithms is very important aspect to improve accuracy and efficiency. Earlier approaches [6-8] were mainly focused on the variation of inertia weight to increase the efficiency of PSO. However, they normally used fixed acceleration constants $C_1$ and $C_2$ ($C_1 = C_2 = 2\ for\ most\ cases$) during the course of iteration.

In this work, instead of using fixed acceleration constants, a weight based acceleration variables $C_{1wP}^t$ and $C_{2wG}^t$is proposed that depends on the current best particle solution and global best particle solution respectively. These variables are updated adapted themselves depending on difference between best and worst fitness value. The velocity update rule for modified PSO can be written as according to following equations

$$V_{i,d}^{t+1} = V_{i,d}^t * \omega^t + C_{1wP}^t * rand(1,d) \odot \left(P_{best,i,d}^t - X_{i,d}^t\right) + C_{1wG}^t * rand(1,d) \odot \left(G_{best,d}^t - X_{i,d}^t\right) \quad (7)$$

Where $C_{1wP}^t$ and $C_{2wG}^t$ are defined as following way

$$C_{1wP}^t = \frac{f(P_{best,i,d}^t) - f_{Worst}^t}{fitness_{Best}^t - fitness_{Worst}^t} \quad (8)$$

$$C_{2wG}^t = \frac{f(G_{best,d}^t) - f_{Worst}^t}{f_{Best}^t - f_{Worst}^t} \quad (9)$$

Where $f(P_{best,i,d}^t)$ , $f(G_{best,d}^t)$ are fitness values corresponding to the local best for *i*-th particle and global best solution respectively. $f_{Best}^t$ and $f_{Worst}^t$ are the best and worst fitness for all particles in current iteration. For a function minimization problem, $f_{Best}^t$ and $f_{Worst}^t$ are defined as follows

$$f_{Best}^t = Minimum\ fitness\ \ for\ X_{1,d}^t, X_{2,d}^t, \dots\dots.., X_{N,d}^t \quad (10)$$

$$f_{Worst}^t = Maximum\ fitness\ for\ X_{1,d}^t, X_{2,d}^t, \dots\dots.., X_{N,d}^t \quad (11)$$

The particles are moving to the optimal point based on above self adaptive mechanism. However, if all particles move to an optimal point during iteration, minimum and maximum fitness value will be same. In that case, $C_{1wP}^t$ and $C_{2wG}^t$ will be undefined. To avoid such case during iteration, we introduce a condition that when $f_{Best}^t$ and $f_{Worst}^t$ are different, $C_{1wP}^t$ and $C_{2wG}^t$ are updated according to equation 8 and 9 else $C_{1wP}^t = C_{2wG}^t = 2$ will be considered.

Moreover, $\omega^t$ is also updated according to LDIW technique i.e. equation 6. The pseudo code of Modified Particle Swarm Optimization (MPSO) algorithm is given as

```
**Start MPSO Algorithm**
Define $N, d, t_{max}, X_{max}, X_{min}, \omega_{max}, \omega_{min}$ and objective function;
for i=1 to $N$ (number of particles)
  Initialize $X_{i,d}$ and $V_{i,d}$;
  $P_{best,i,d} = X_{i,d}$;
end;
Evaluate fitness value $f(X_{i,d})$for all $N$ particle position (i=1, 2,...., $N$);
G_best,d= Min ($f$);
Assign value of $\omega^t$ according to LDIW [Equation 6];
While ( $t \le t_{max}$)
  for i=1 to $N$
        $f_{Best}^t$ = Min ($f$);
        $f_{Worst}^t$= Max ($f$);
        Evaluate $f(P_{best,i,d}^t)$, $f(G_{best,d}^t)$;
        if ($f_{Best}^t \ne f_{Worst}^t$)
            Calculate $C_{1wP}^t$ and $C_{2wG}^t$ using equation 8
            and 9 respectively;
        else
            $C_{1wP}^t$=$C_{2wG}^t$=2;
      end if;
      Update the velocity $V_{i,d}$using equation 7;
      Update the position $X_{i,d}$using equation 2;
      Evaluate fitness value for $f(X_{i,d})$;
       if $f(X_{i,d}) < f(P_{best,i,d}^t)$
          $P_{best,i,d}^t$=$X_{i,d}$;
       end if;
       if $f(P_{best,i,d}^t) < f(G_{best,d}^t)$
          $G_{best,d}^t = P_{best,i,d}^t$;
       end if;
  end for;
  $X^* = G_{best,d}^t$;
end while;
Return $X^*$;
**End MPSO Algorithm**
```

## IV. Experrimantal Results

To validate the proposed MPSO algorithm, we have chosen 6 different benchmark functions which are Ackley, De Jong, Easom, Easom, Griewank, Rastrigin and Rosenbrock function respectively. All of these functions are multimodal except De Jong function (also known as square function) which is unimodal in nature. The details of these functions, respective search ranges of variables and corresponding global minima points are shown in table 2. We have considered 10 dimensional optimization problem (*d*=10) for all functions.

### *A. Parameters Setting*

In this work, CIWPSO, RIWPSO, LDIWPSO and proposed MPSO were applied for above mentioned function minimization problems. Performance of MPSO is also compared with others for validation purpose. For this, a swarm of 50 particles and 1000 maximum iteration was taken. Following table shows values of respective different inertia weight parameters and acceleration constants for different techniques. All the techniques were implanted using Matlab 7.6 with 2 GB RAM, Dual Core processor and Windows7 operating System.

### *B. Analysis of Results*

In this research work, all techniques are executed for all benchmark functions and result analysis is done based on five different criteria i.e. best fitness, average fitness, worst fitness, average execution time and convergence speed.

As the performance of PSO is also depending on initialization of particles, therefore, each program for each function is simulated for 50 times with different initialization. The best and worst fitness is the minimum and maximum fitness value respectively among those 50 optimal outputs corresponding to 50 times run. Average fitness and execution time is the mean of all cases of output fitness and execution time respectively.

Table 3 shows a comparative study among different strategies for all six benchmark problems on the basis of best fitness, average fitness, worst fitness and average execution time where best values are shown in bold letter and worst values are denoted by inverted letter.

Next, we have observed the convergence of each algorithm for each of the functions. Followings figures depicts how the fitness value decreases with respect to iteration number for all strategies and functions.

From Table 3, it can be observed that proposed MPSO gives best fitness for all function except De Jong and Easom functions. Though the best fitness for De Jong is quite satisfactory, but MPSO underperformed for Easom function. MPSO has been stuck to a local minima point ($f_*$=0) for Easom function. LDIWPSO performs better with respect to average fitness and worst fitness while CIWPSO is the fastest technique on the basis of average execution time. However, execution time of MPSO is slightly higher than others due to incorporation of self adaptive technique in the algorithm. Now, if we observe the convergence graph for different algorithm, we observed that MPSO converge faster than other methods for all functions except Easom. RIWPSO is the worst technique on the basis all performance parameters except average execution time.

Table 4 summarizes the output of this study where best and worst techniques for different parameters are mentioned. Best technique is selected on the basis of maximum voting for each parameter against all functions. However, best fitness and convergences are most important parameters for an optimization technique. It can be observed that MPSO is better technique on the basis of best fitness and convergence. So, MPSO is preferable than others technique such as LDIWPSO, CIWPSO and RIWPSO.

Table 1. Different Benchmark functions

| Function name | Function | Range of search | Global minima point |
|---|---|---|---|
| **Ackley** | $-20\exp\left[\frac{1}{5}\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right] - \exp\,[\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)] + 20 + e$ | $-30 \le x_i \le 30$ | $f_*$= 0 at (0,0,…0) |
| **De Jong** | $\sum_{i=1}^{d}x_i^2$ | $-5.12 \le x_i \le 5.12$ | $f_*$= 0 at (0,0,…0) |
| **Easom** | $(-1)^{d+1}\prod_{i=1}^{d}\cos(x_i)\ \ \exp[-\sum_{i=1}^{d}(x_i-\pi)^2]$ | $-30 \le x_i \le 30$ | $f_*$= -1 at $(\pi, \pi, \ldots, \pi)$ |
| **Griewank** | $\frac{1}{400}\sum_{i=1}^{d}x_i^2 - \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$ | $-600 \le x_i \le 600$ | $f_*$= 0 at (0,0,…0) |
| **Rastrigin** | $100d + \sum_{i=1}^{d}[x_i^2 - 10\cos(2\pi x_i)]$ | $-5.12 \le x_i \le 5.12$ | $f_*$= 0 at (0,0,…0) |
| **Rosenbrock** | $\sum_{i=1}^{d-1}[(x_i-1)^2 + 100(x_{i+1}-x_i^2)^2]$ | $-5 \le x_i \le 5$ | $f_*$= 0 at (1,1,…,1) |

Table 2. Optimization Parameters for different inertia

| Techniques | Inertia Weight | Acceleration Constant |
|---|---|---|
| **CIWPSO** | $\omega^t = 0.5$ | $c_1 = c_2$ = 2 |
| **RIWPSO** | $\omega^t = 0.5 + rand/2$ | $c_1 = c_2$ = 2 |
| **LDIWPSO** | $\omega^t = \omega_{max} - \left\{\frac{\omega_{max}-\omega_{min}}{t_{max}}\right\} \times t$ where $\omega_{max} = 0.9$ and $\omega_{min} =$ 0.4 | $c_1 = c_2$ = 2 |
| **MPSO** | $\omega^t = \omega_{max} - \left\{\frac{\omega_{max}-\omega_{min}}{t_{max}}\right\} \times t$ where $\omega_{max} = 0.9$ and $\omega_{min} =$ 0.4 | Self-Adaptive [Eq. 8 and 9] |

Table 3. Comparative study for different PSO algorithm

| **Algorithm** | **Function name** | **Ackley** | **De Jong** | **Easom** | **Griewank** | **Rastrigin** | **Rosenbrock** |
|---|---|---|---|---|---|---|---|
| **CIWPSO** | Best fitness | 4.44E-15 | **1.34E-58** | **-1** | *0.352708* | 0.994981 | 4.03E-05 |
| **RIWPSO** | | *0.100554* | *0.421686* | -0.99951 | 0.053163 | *11.29862* | *5.271942* |
| **LDIWPSO** | | 1.64E-13 | 1.09E-28 | **-1** | 0.018774 | 1.000377 | 8.87E-05 |
| **MPSO** | | **7.99E-15** | 1.32E-29 | *-4E-13* | **0.014455** | **2.13E-07** | **2.42E-05** |
| **CIWPSO** | Average fitness | 0.079133 | **1.26E-27** | **-1** | *2.25284* | **4.803865** | 6.403575 |
| **RIWPSO** | | *3.631267* | *2.63E-05* | -0.35416 | 0.762569 | 32.43858 | *109.912* |
| **LDIWPSO** | | **0.000132** | 4.67E-13 | **-1** | **0.089828** | 4.826136 | **5.589858** |
| **MPSO** | | 0.028382 | 7.84E-17 | *-8E-15* | 0.137153 | 8.121388 | 6.935556 |
| **CIWPSO** | Worst fitness | 1.646224 | **4.3E-26** | -0.99996 | *11.59614* | 19.89914 | 96.85403 |
| **RIWPSO** | | *13.11975* | *6.714779* | -2E-21 | 4.920912 | *80.68314* | *681.5958* |
| **LDIWPSO** | | **0.006591** | 2.34E-11 | **-1** | **0.366723** | **11.93968** | **28.53141** |
| **MPSO** | | 1.155319 | 3.39E-15 | *-1.2E-87* | 0.50223 | 17.90925 | 82.39822 |
| **CIWPSO** | Average Execution Time (Sec) | **30.25599** | **10.38025** | 29.75094 | 29.65551 | **12.11045** | 11.20735 |
| **RIWPSO** | | 30.56758 | 10.40836 | 30.12699 | 29.53889 | 12.38601 | **11.09225** |
| **LDIWPSO** | | 30.5915 | 10.82977 | **28.99117** | **29.01396** | 12.636369 | 11.626796 |
| **MPSO** | | *131.2823* | *71.70359* | *128.9571* | *131.992* | *85.586803* | *81.651894* |

Table 4. Performance Analysis of different algorithm

| **Performance Parameter** | **Best Techniques** | **Worst Technique** |
|---|---|---|
| Best fitness | MPSO | RIWPSO |
| Average fitness | LDIWPSO | RIWPSO |
| Worst fitness | LDIWPSO | RIWPSO |
| Average Execution Time | CIWPSO | MPSO |
| Convergence | MPSO | RIWPSO |

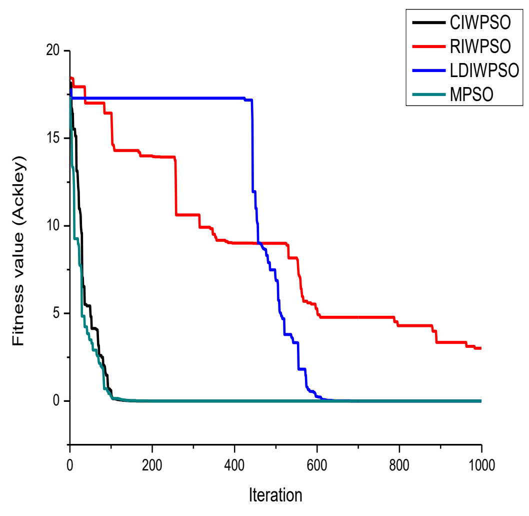

Fig.1. Convergence for Ackley function

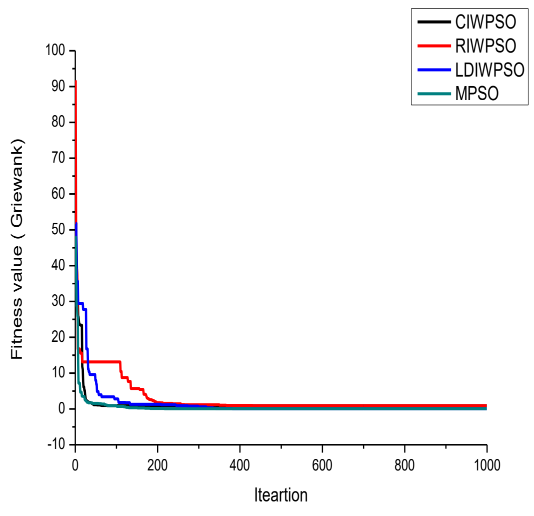

Fig.4. Convergence for Griewank function

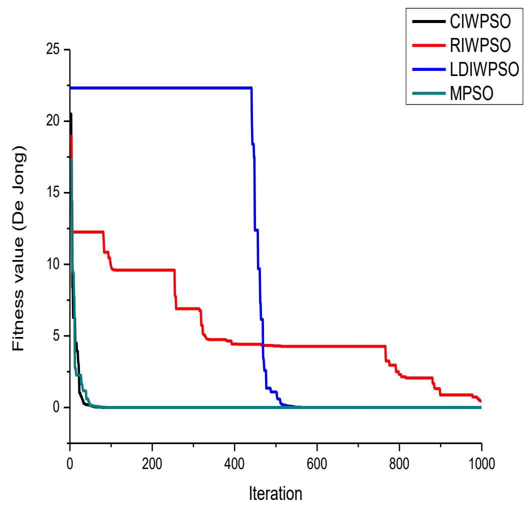

Fig.2. Convergence for De Jong function

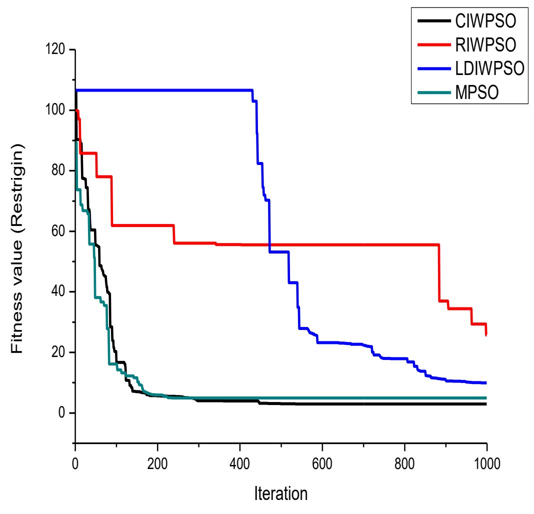

Fig.5. Convergence for Restrigin function

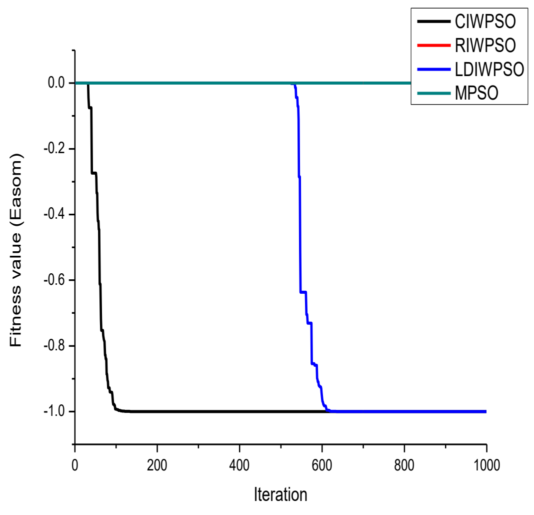

Fig.3. Convergence for Easom function

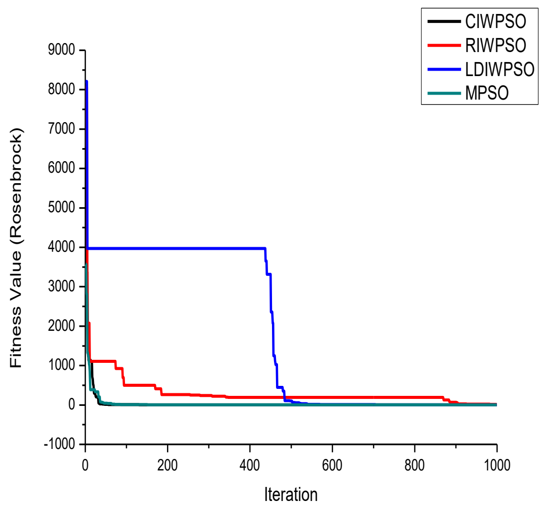

Fig.6. Convergence for Rosenbrock function

## V. Conclusion

Several state of art techniques for inertia weight variation of PSO are already developed to increase the efficiency of PSO. However, in this paper, a Modified Particle Swarm Optimization Algorithm based on self-adaptive acceleration constants is proposed for further improvement in the accuracy of PSO. Here, acceleration constants depend on the global best particle solution and current best particle solution and updated themselves during the course of iteration. The proposed method is tested against six different benchmark functions. The results are also compared with other three strategies i.e. CIWPSO, RIWPSO and LDIWPSO. It is found that MPSO is most suitable for function optimization with best fitness value and convergence. However, its execution time is slightly higher than others technique. In future, different strategies for updating acceleration constants may be employed for further improvement of the performance of PSO.

## References

[1] J. Kennedy and R.C. Eberhart, "Particle swarm optimization," in *IEEE International Conference on Neuran Networks*, pp. 1942–1948, 1995.

[2] A. Nickabadi, M. M. Ebadzadeh, and R. Safabakhsh, "A novel particle swarm optimization algorithm with adaptive inertia weight," *Applied Soft Computing*, vol. 11(4), pp. 3658-3670, 2011.

[3] M. A. Arasomwan and A. O. Adewumi, "On the Performance of Linear Decreasing Inertia Weight Particle Swarm Optimization for Global Optimization," *The Scientific World Journal*, vol. 2013, Article ID 860289, pp. 1-12, 2013.

[4] Y. H. Shi and R.C. Eberhart, "A modified particle swarm optimizer," in *IEEE International Conference on Evolutionary Computation*, pp. 69–73. 1998.

[5] R. C. Eberhart and Y. H. Shi, "Tracking and optimizing dynamic systems with particle swarms," in: *Congress on Evolutionary Computation*, 2001.

[6] R. C. Eberhart and Y. H. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *IEEE Congress on Evolutionary Computation*, pp. 84–88, 2000.

[7] Y. H. Shi and R. C. Eberhart, "Experimental study of particle swarm optimization," in *SCI2000 Conference*, 2000.

[8] J. Xin, G. Chen, and Y. Hai., "A Particle Swarm Optimizer with Multistage Linearly-Decreasing Inertia Weight," in *IEEE International Joint Conference on Computational Sciences and Optimization (CSO-2009)*, pp. 505–508, 2009.

[9] 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.

[10] R. F. Malik, T. A. Rahman, S. Z. M. Hashim, and R. Ngah, "New Particle Swarm Optimizer with Sigmoid Increasing Inertia Weight," *International Journal of Computer Science and Security*, vol. 1(2), pp. 35-44, 2007.

[11] Y. Feng, G. F. Teng, A. X. Wang, and Y.M. Yao., "Chaotic Inertia Weight in Particle Swarm Optimization," in *Second IEEE International Conference on Innovative Computing, Information and Control. ICICIC-07*, pp. 475-475, 2007.

[12] K. Kentzoglanakis and M. Poole., "Particle swarm optimization with an oscillating Inertia Weight," in *11th Annual conference on Genetic and evolutionary computation*, pp 1749–1750, 2009.

[13] M. S. Arumugam and M. V. C. Rao, "On the performance of the particle swarm optimization algorithm with various Inertia Weight variants for computing optimal control of a class of hybrid systems," *Discrete Dynamics in Nature and Society*, vol. 2006, Article ID 79295, pp. 1-17, 2006.

[14] W. Al-Hassan, M. B. Fayek, and S.I. Shaheen, "PSOSA: An optimized particle swarm technique for solving the urban planning problem," in *The IEEE International Conference on Computer Engineering and Systems*, pp 401–405, 2006.

[15] H. R. Li and Y.L. Gao, "Particle Swarm Optimization Algorithm with Exponent Decreasing Inertia Weight and Stochastic Mutation," in *The IEEE Second International Conference on Information and Computing Science*, pp. 66–69, 2009.

[16] G. Chen, X. Huang, J. Jia, and Z. Min, "Natural exponential Inertia Weight strategy in particle swarm optimization", in *The IEEE Sixth World Congress on Intelligent Control and Automation (WCICA-2006)*, pp. 3672–3675, 2006.

[17] Y. Gao, X. An, and J. Liu., "A Particle Swarm Optimization Algorithm with Logarithm Decreasing Inertia Weight and Chaos Mutation", in *IEEE International Conference on Computational Intelligence and Security (CIS'08)*, pp. 61–65, 2008.

[18] A. Chatterjee and P. Siarry, "Nonlinear inertia weight variation for dynamic adaption in particle swarm optimization," *Computer and Operations Research*, vol. 33 pp. 859–871, 2006.

[19] A. Khan, S. Mandal, R. K. Pal, and G. Saha, "Construction of Gene Regulatory Networks Using Recurrent Neural Networks and Swarm Intelligence," *Scientifica*, vol. 2016, Article ID 1060843, pp. 1-14, 2016.

[20] N. C. Bansal, P. K. Singh, M. Saraswat, A. Verma, S. S. Jadon, and A. Abraham, "Inertia Weight strategies in Particle Swarm Optimization," in *Third World Congress on Nature and Biologically Inspired Computing (NaBIC'11)*, pp. 633-640, 2011.

[21] S. Mirjalili, S.M. Mirjalili, and A.Lewis, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, pp.46-61, 2014.

[22] M. Dorigo, M. Birattari, and T. Stutzle,. Ant colony optimization. IEEE computational intelligence magazine, 1(4), pp.28-39. 2006.

[23] D. Karaboga, B. Gorkemli, , C. Ozturk, and N. Karaboga, "A comprehensive survey: artificial bee colony (ABC) algorithm and applications," *Artificial Intelligence Review*, vol. 42(1), pp.21-57, 2014.

[24] X. S. Yang, "A new metaheuristic bat-inspired algorithm," *Nature inspired cooperative strategies for optimization (NICSO 2010)*, pp. 65-74, 2010.

[25] X. Yingwei, "Research on SVG DC-Side Voltage Control Based-on PSO Algorithm," *International Journal of Information Technology and Computer Science*, vol. 8(10), pp.29-38, 2016. DOI: 10.5815/ijitcs.2016.10.04

[26] F. S. Milani and A. H. Navin, "Multi-Objective Task Scheduling in the Cloud Computing based on the Patrice Swarm Optimization," *International Journal of*

*Information Technology and Computer Science*, vol. 7(5), pp.61-66, 2015. DOI: 10.5815/ijitcs.2015.05.09

[27] A. Verma and S. Kaushal, “Cost Minimized PSO based Workflow Scheduling Plan for Cloud Computing”, *International Journal of Information Technology and Computer Science*, vol. 7(8), pp. 37-43, 2015. DOI: 10.5815/ijitcs.2015.08.06

[28] A. Babazadeh, H. Poorzahedy and S. Nikoosokhan, “Application of particle swarm optimization to transportation network design problem,” *Journal of King Saud University-Science*, vol. 23(3), pp.293-300, 2011.

## Author’s Profile

**Sudip Mandal** received M.Tech. degree in Electronics and Communication Enigneering from Kalyani Govt. of Engineering College on 2011. Now, he hold the position of Head of Electronics and Communication Engineering Department in Global Institute of Management and Technology, Krishnanagar, India. He is also pursuing Ph.D. degree from University of Calcutta. His current research work includes Computational Biology, Optimization, Soft Computing and Tomography. He is Editorial Member for Global Journal on Advancement in Engineering & Science (GJAES) and Computer Applications: An International Journal (CAIJ) under AIRCCSE. The author is also member Computational Intelligence Society and Man, System & Cybernetics Society of IEEE. The author published 22 International & National journal and conference papers so far.

**How to cite this paper:** Sudip Mandal,"A Modified Particle Swarm Optimization Algorithm based on Self-Adaptive Acceleration Constants", International Journal of Modern Education and Computer Science(IJMECS), Vol.9, No.8, pp.49-56, 2017.DOI: 10.5815/ijmecs.2017.08.07