

Velocity Restriction based Improvised Particle Swarm Optimization Algorithm

Mouna.H*, Mukhil Azhagan.M.S, Radhika M.N,
Mekaladevi. V**, Nirmala Devi. M
Department of Electronics and Communication Engineering
Amrita School of Engineering, Coimbatore
Amrita Vishwa Vidyapeetham
Amrita University, India
*mouna.harikumar@gmail.com,** v_mekaladevi@cb.amrita.edu

Abstract— The Particle Swarm Optimization (PSO) Algorithm attempts on the use of an improved range for inertial weight, social and cognitive factors utilizing the pareto principle. The function exhibits better convergence and search efficiency than the classic PSO algorithms that use conventional linearly varying or exponentially varying inertial weights. It also exhibits a technique to intelligently navigate the search space around the obtained optima and look for better optima if available and continue converging with the new values using a velocity restriction factor based on the Pareto principle. The improvised algorithm searches the neighborhood for the global optima while maintaining frequent resets in the position of some particles in the form of a mutation based on its escape probability. The choice of the social and the cognitive parameters depend on the function used and the type of convergence required, and has the flexibility to pick the neighborhood range for the search. The results have been compared and tabulated against popular PSO with conventional weights and it has been shown that the improvised PSO performs much better on various benchmark functions.

Index Terms- Particle Swarm Intelligence; Global Optimization; Intelligent Search; Inertial Weight; Velocity Restriction; Pareto Principle;

I.INTRODUCTION

Particle swarm optimization (PSO) is a computational algorithm that simulates natural swarm behavior most notably found in certain species of birds, fish and bees. The algorithm was introduced by Kennedy and Eberhart in 1995[1-2]. The particles in a swarm, analogous to birds in a flock or fish in a school, move around the search space in a predefined pattern that is close to the natural search of a bird for its food in search of the global optima or in the natural case, the food. The communication between members of a flock is also simulated and each particle in the swarm is aware of its own optima and the best optima among the swarm. Different particles explore in different velocities and from different positions, all of which are randomized to obtain results close to the natural order of swarming [3].

PSO being faster in terms of iterations and processing time has progressed rapidly in recent years, and has been used for problems in Artificial intelligence, Material design and various fields that require a quick optimization [4-5]. Conventional PSO gets stuck at a local minimum [6]. Research has been on trying to accelerate the convergence speed and achieving a better accuracy [7-9]. Reported literature has either a linearly decreasing pattern alone or a combination of two

different algorithms, popularly known as hybrid algorithms, which are comparatively slow on convergence[10-12].The improvised algorithm attempted in this paper has a varying *inertial weight* based on the Pareto principle, thus enabling it to converge to a better value with more precision.

To achieve the above mentioned superiority over the other algorithms, three techniques have been introduced. The first technique, the *velocity restriction* is based on the Pareto principle (or the 80-20 rule)[16-17].The second technique is on a mutation based term called the *escape probability*, this allows for a way of doing extensive exploration that focuses on finding other local optima easily. The third technique uses an improved *inertial weight* [10], which will progressively converge the search towards the minima over higher iterations. The designer is allowed to set the cognitive and social parameters, so the user has control over the neighbor of convergence, either towards the *pbest* or *gbest* depending on the requirement of the algorithm. At each iteration, there is a condition for mutation that is based on its *escape probability*, which is the number of times it moves out of the boundary. This serves a dual purpose of maintaining the position within the search space of interest and also mutates the position of the particle frequently. This frequency is controlled by the *escape probability*, which is in turn controlled by the initial velocity that is set by the designer. Combining the three techniques - *mutation*, *velocity restriction*, and the refined *inertial weight*-both based on the pareto principle-the algorithm has been made adaptive and intelligent to work with varied functions. Simulation results have been produced to show better convergence and precision.

II. PARTICLE SWARM OPTIMIZATION (PSO)

The PSO algorithm was first proposed by considering a swarm of particles of size N [1-2].The best optimum position in the search space was found by these swarm of particles using their swarm intelligence. The conventional PSO uses five basic principles namely Quality principle, Proximity principle, stability principle, Diverse response principle, and adaptability principle [1].At the start of the PSO, the number of variables D is initialized and the objective function f is specified. The required parameters such as population size, swarm size, total number of iterations $itmax$, cognitive factor c_1 , social factor c_2 and *inertial weight* ω are initialized. The independent variables are given their boundary conditions in which they can search for the best optimum position. The random values of position and velocity are initialized as the *pbest* values for each particle, and *gbest* value for the swarm. *pbest* and *gbest* are the personal best of each particle and global best of the swarm respectively. In the given search space, the new position value is found by each of the particles.

The position and velocity are calculated using the formulae (1), (2) given below.

$$v_i^d = \omega * v_{i-1}^d + c_1 * rand_1^d * (pbest_i^d - x_{i-1}^d) + c_2 * rand_2^d * (gbest^d - x_{i-1}^d)$$

(1)

$$x_i^d = x_{i-1}^d + v_i^d \tag{2}$$

Where v_i^d is the velocity of the current iteration for each argument, ω is the *inertial weight* [10]. $rand_1^d$ and $rand_2^d$ are two random distributions that range between 0 and 1. x_i^d is the position

of each particle that will be updated from its previous value.

The new position value is compared and checked with its previous value. If the new position value is found to be better than the previous value, the *pbest* value is updated else the previous value is retained. The best value among the *pbest* of all the particles is taken, if the value is better than the previous *gbest*, then it is replaced as the *gbest* value, else the older value is retained. Inertia weight plays an important role in PSO, by significantly affecting the accommodation between exploitation and exploration in the PSO process. Different variants of PSO can be obtained by changing parameters such as the cognitive and social factor, different inertia weights, swarm size, network topologies in PSO etc [13]. Hybridization and multi-objective are some of the variants. In Hybridization, for example, Genetic Algorithm and PSO can be combined, GA's mutation technique can be combined with PSO to prevent PSO from getting stuck at a local optima [14].

III.IMPROVISED PSO

The difference between conventional PSO [1-2] and the improvised PSO is that it has techniques for *mutation*, *velocity restriction* and improvised ranges for factors. This PSO works just like the conventional PSO except for the fact that it has a restricted search in different range of weights, and *velocity restriction* based on the pareto principle. The Pareto principle states that 20 percent of the cause is responsible for 80 percent of the outcome or vice versa, depending on the perspective. Applying this, the *inertial weight*, which is calculated using formula (3) is varied from values between 0.8 to 0, to cover 80 percent of the area, that is a higher neighborhood of exploration, to find the 20 percent of the local optima after convergence has started.

$$\omega_i = \omega_{max} - (\omega_{max} - \omega_{min}) * (it / (itmax)) \quad (3)$$

Where ω_i is the weight of each iteration, ω_{max} is around 0.8 and ω_{min} is around 0. Values taken in the trials are 0.7 and 0.1 respectively. *itmax* denotes the maximum number of iterations considered and *it* denotes the current iteration. Also, a technique of *velocity restriction*, calculated using the formula (4), that modifies the effect of the previous velocity on the existing position, is included.

$$V_r = e^{(-it / (k * itmax))} \quad (4)$$

Where V_r is the *velocity restriction* factor. k is a constant, whose value is taken as 4 for an optimum range. Equation (4) should be multiplied along with velocity during every iteration to restrict its boundary. It decreases exponentially, and the speed can be modified by changing the value of k by the user depending on the need. Frequent mutations are performed to help the exploration process, determined by an *escape probability*. This *escape probability* is calculated using an algorithm that also prevents the particle to move out of the boundary, thus stabilizing the swarm search. The Pseudo code for the improvised PSO is given below:

PSEUDOCODE

Step a: Initialize the number of dimensions, all the required parameters (such as the size of population and swarm, total number of iterations $itmax$, cognitive and social factors and *inertial weights*), and specify the objective function.

Step b: Calculate the *inertial weight* for each iteration using (3), within the range 0.8 - 0.0.

Step c: Calculate the *velocity restriction* parameter for each of the iterations using (4)

Step d: Define the boundary conditions in which the particles search for the best optimum position.

Step e: Initialize random values for Position and velocity of each particle as $pbest$ and $gbest$.

Step f: Find the next $gbest$ and $pbest$ values.

Step g: Find the new position value of the particle

Step h: Find the new velocity of each particle using *velocity restriction* technique using (1) and (4).

Step i: Each individual position should be updated using (2) and the new velocity.

Step j: Restrict each particle to the defined boundary using the mutation technique, utilizing *escape probability*.

Step k: Compare and check the new position value with the previous value. If the new position value obtained is better than the previous position value, go to m , else go to n .

Step l: Update the previous values of $pbest$ and $gbest$ with the new best value obtained.

Step m: Go to *Step o*.

Step n: Retain the previous values of $pbest$ and $gbest$.

Step o: If $i \leq itmax$, $i++$, go to step g else go to step p .

Step p: Indicate the optimum value of $pbest$ and $gbest$.

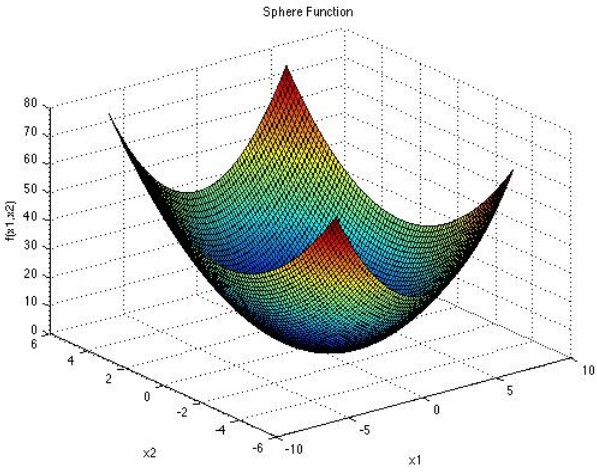
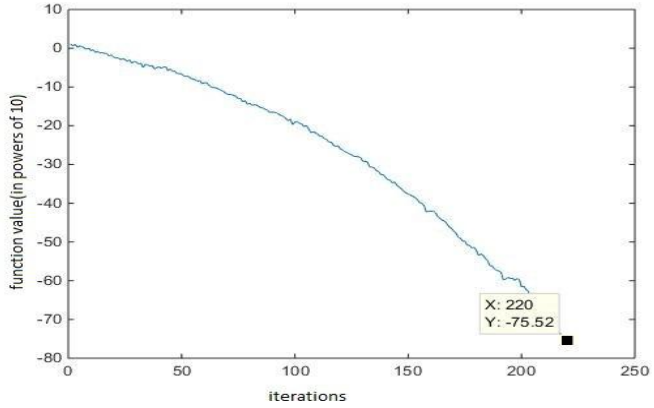
Figure I: Pseudo code for the improvised PSO.

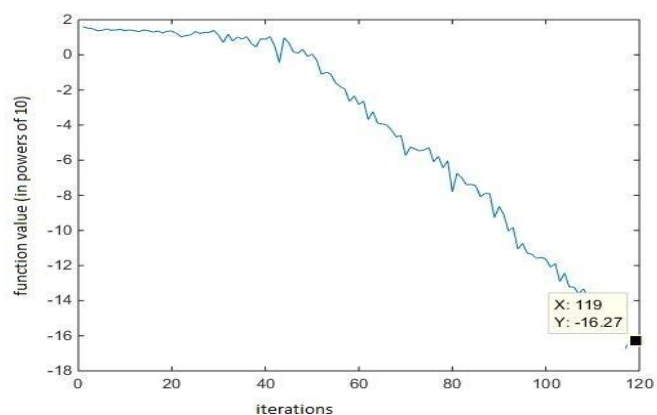
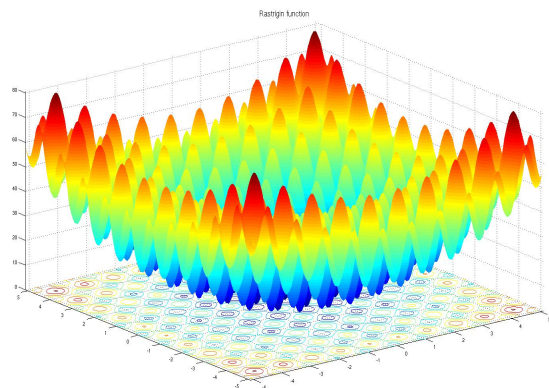
V.SIMULATION RESULTS AND ANALYSIS

The improvisation performed on Particle Swarm Optimization (PSO) was tested on benchmark functions and the results acquired are improved comparatively [10-12]. The Code was simulated in Matlab® used is a PC with core I3 4005u, 1.7GHz and 4 GB RAM. Benchmark functions used are (a) Sphere function, which is continuous, unimodal and has D local minima (b) Rastrigin function, which is multimodal and has several local minima (c) Rosenbrock function, also known as valley or banana function, is unimodal (d) Michalewicz function, which is multimodal and has D local minima and are usually referred as valleys and ridges (e) Shubert function, which has many local and global minima.

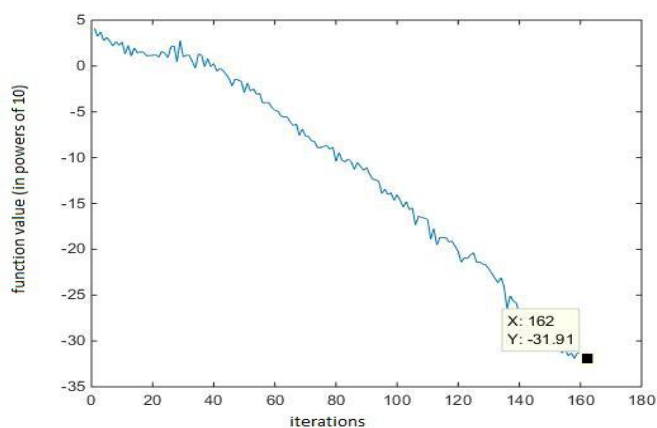
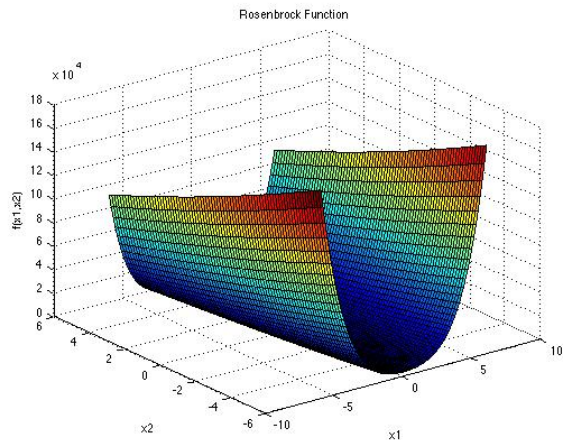
The results have been obtained for 220 iterations and over 30 independent trials with 100 particles. The search has been done over a search space of $[-5.12, 5.12]$ for (a), (b), (c) and (e) functions and $[-\pi, \pi]$ for (d), as given in [18]. *Table 1* shows the 2D plot between mean function value of all particles and number of iterations of Sphere, Rastrigin, Rosenbrock, Michalewicz and Shubert functions respectively including equations and 3D plots.

Table1: Equations, 3D and 2D Plot for various Benchmark Function.

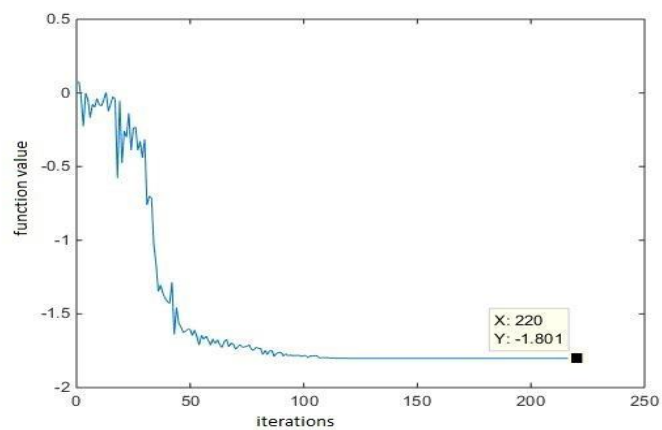
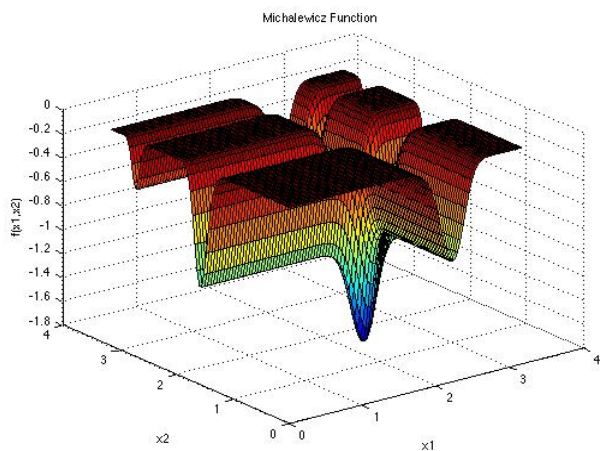
(a) Sphere Function:	(5)
	
(b) Rastrigin Function:	(6)



(c) *Rosenbrock Function:* (7)



(d) *Michalewicz Function:* (8)



(e) Shubert Function: (9)

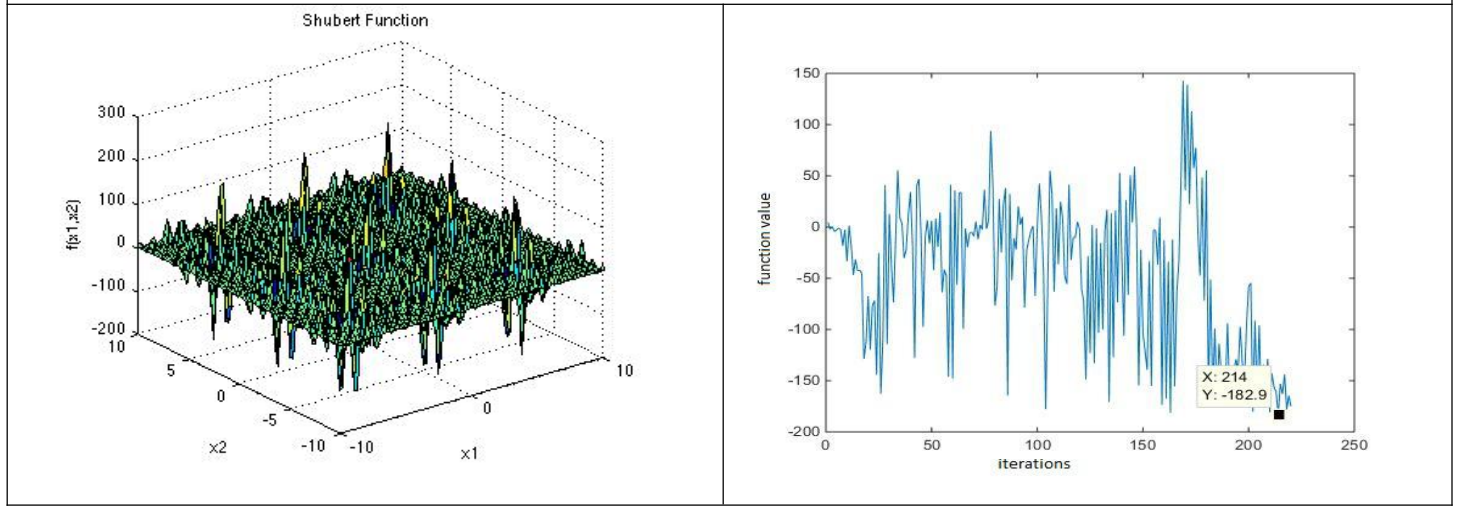


Table 1 shows plots for the benchmark functions , for (a),(b),(c)the y-axis for the 2D plot is indexed in powers of 10 ,as the expected values from mathematical calculation [18] is close to zero . For (d), (e) the y-axis is in real values. Table 2 shows the iteration at which the minimum value is obtained for each benchmark function. Combining 2d plots from Table 1 and Table 2, various inferences can be made. In Table 1 for the Shubert function , which is multimodal , a large number of spikes can be seen , which denote various particles exploring other parts of the search space for potential global minima until the 220 iterations end, whereas the minima has been reached at around the 26th iteration Table 2 . This demonstrates the ability of the algorithm to explore exhaustively even after getting settled at the minima, due to the *mutation* factor introduced using *escape probability*. For the sphere function , which has a single minima , the algorithm tries to obtain the best possible value , from Table 3 , it can be seen that the algorithm is precise up to 10^{-71} on an average , In such functions , the algorithm is able to choose exploitation over exploration thus leading to much better values as proven in Table 4 .

Table 2: Iterations at which Benchmark functions converge.

Function	Mean	Std.Dev.	Fastest	Slowest
Sphere	220	220	220	220
Rastigin	88.03333	6.895692	78	109
Shubert	26.2	2.265179	23	31
Michelawicz	30.03333	3.090400413	26	40
Rosenbrock	136.5333	5.309285	128	148

In Table 3 , X_1 denotes the position in the first dimension. X_2 denotes the position in the second dimension and $fgbest$ is the fitness value , that is dependant on X_1 and X_2 as described by

equations in Table 1 .Table 3 shows the values of X_1, X_2 and $fgbest$ using the equations (5), (6), (7), (8) and (9) for various benchmark functions.

Table 3: Results Obtained of the algorithm on various benchmark functions.

Benchmark Functions		Mean	Std.Dev.	Best Fit	Worst Fit	Expected value [18]
Sphere	X_1	-1.45E-37	8.87989E-37	6.87E-37	-4.79E-36	0
	X_2	-5.60E-37	3.49677E-36	2.30E-36	-1.89E-35	0
	$Fgbest$	1.29E-71	6.54056E-71	1.23E-82	3.58E-70	0
Rastrigin	X_1	-1.63E-09	2.37965E-09	-3.75E-09	3.51E-09	0
	X_2	-4.79E-10	2.03184E-09	-3.73E-09	2.89E-09	0
	$Fgbest$	0	0	0	0	0
Shubert	X_1	-0.67279	1.533821	-1.42662	4.858057	Several Minima
	X_2	-0.29594	2.076847	-1.42523	4.858044	Several Minima
	$Fgbest$	-186.725	0.007744	186.7304	-186.705	-186.7309
Michalewicz	X_1	2.202906	5.87319E-10	2.202906	2.202906	2.20
	X_2	1.570796	2.11439E-09	1.570796	1.570796	1.57
	$Fgbest$	-1.8013	6.77522E-16	-1.8013	-1.8013	-1.8013
Rosenbrock	X_1	1	0	1	1	1
	X_2	1	0	1	1	1
	$Fgbest$	0	0	0	0	0

In Table 3 , the best fit column denotes the best value obtained ,this value is the global minimum that has been obtained through the algorithm over the trials .The mean and standard deviation denote the algorithms variation from the bestfit value. It has been shown that , the algorithm performs with minimal variance for most of the benchmark functions when the $fgbest$ value is concerned . Incase of X_1 and X_2 , the average and the standard deviation are close to the expected values ,except in case of (e), the shubert function , as the function exhibits the same minimum value at multiple points in the given search space .This discrepancy is expected out of such a function and can verified mathematically [18] .

Table 4 shows the $fgbest$ values using the introduced algorithm i.e the fitness value and it is compared with [11], [12], [15]. The proposed algorithm exhibits much better values in unimodal functions like Sphere Function, and almost precise values in multimodal functions like the Rastrigin, Rosenbrock, Michalewicz, etc.

Table 4: Performance comparison amongst literature and introduced algorithm on Benchmark Functions.

Function Name		[11]	[12]	[15]	Obtained Results
Mean	Sphere	2.46E-11	1.44E-23	3.80E-27	1.29E-71
	Rastrigin	NA	0.01	0.01	0
	Michalewicz	-1.87688	-1.89473	-1.8966	-1.8013
	Shubert	-186.704	-186.728	-186.717	-186.725
Standard Deviation	Sphere	1.35E-10	7.86E-23	2.08E-26	6.54056E-71
	Rastrigin	0.181654	0.252429	0.252429	0
	Michalewicz	0.093425	0.090563	0.086769	6.77522E-16
	Shubert	0.141864	0.011918	0.076175	0.007744

Table 4 contains the comparison of mean and standard deviation on the benchmark functions such as spherical, rastrigin, michaelwicz and shubert. Rosenbrock function is neglected due to unavailability of information. The best value amongst the compared values is highlighted. On comparison of the mean values with [11] and [15], it is seen that there is 10^{60} increase and a 10^{44} increase respectively for sphere function. For Rastrigin function, the expected value of 0 has been obtained. For Michalewicz function, the expected value has been obtained. For Shubert function, [12] exhibits the best value, and the proposed algorithm is only second to it with an error of 0.0016 %. In Table 3 it has also been shown that the algorithm produces the expected result for rosenbrock function.

VI. CONCLUSION

The PSO variant introduced in the paper has three modifications, namely, a new range of linearly varying inertia weight to work with most real time and natural order functions that follow the exponential rule, a *mutation* technique to control particles that move too fast and increase exploration capability, and a *velocity restriction* factor that converges the search space exponentially over the given range. The Algorithm has been proven to work well for both unimodal and multimodal functions. It can especially tackle multimodal functions better due to the inclusion of the Pareto effect in various phases of the Algorithm. The Algorithm seems to be promising for any number of dimensions with any function and is expected to produce a better solution. Improvement of the order of 10^{40} is seen in spheric function and expected values have been obtained in other benchmark functions

REFERENCES

- [1] J. Kennedy and R. C. Eberhart, Particle swarm optimization, in *Proc. IEEE Int. Conf. Neural Netw.*, Perth, Australia, 1995, vol. 4, pp. 1942-1948.
- [2] J. Kennedy, R. C. Eberhart, and Y. H. Shi, *Swarm Intelligence*. San Mateo, CA: Morgan Kaufmann, 2001.
- [3] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micromachine Human Sci.*, Nagoya, Japan, 1995, pp. 39-43. *Symp. Micromachine HumanSci.*, Nagoya, Japan, 1995, pp. 39-43.
- [4] R. C. Eberhart and Y. H. Shi, "Particle swarm optimization: Developments, applications and resources," in *Proc. IEEE Congr. Evol. Comput.*, Seoul, Korea, 2001, pp. 81-86.
- [5] G. Ciuprina, D. Ioan, and I. Munteanu, "Use of intelligent-particle swarm optimization in electromagnetics," *IEEE Trans. Magn.*, vol. 38, no. 2, pp. 1037-1040, Mar. 2002.
- [6] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Trans. Evol. Comput.*, vol. 10, no. 3, pp. 281-295, Jun. 2006.
- [7] S.-Y. Ho, H.-S. Lin, W.-H. Liah, and S.J. Ho, "OPSO: Orthogonal particle swarm optimization and its application to task assignment problems," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 38, no. 2, pp. 288-298, Mar. 2008.
- [8] B. Liu, L. Wang, and Y. H. Jin, "An effective PSO-based memetic algorithm for flow shop scheduling," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 37, no. 1, pp. 18-27, Feb. 2007.
- [9] R. C. Eberhart and Y. Shi, "Guest editorial," *IEEE Trans. Evol. Comput. Special Issue Particle Swarm Optimization*, vol. 8, no. 3, pp. 201-203, Jun. 2004.
- [10] Zhi-Hui Zhan and Jun Zhang, "Adaptive Particle Swarm Optimization," in *IEEE Transactions on systems, man, and cybernetics-Part b: Cybernetics*, Vol. 39, No. 6, December 2009
- [11] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *Proc. IEEE World Congr. Comput. Intell.*, 1998, pp. 69-73.
- [12] Ting-Yu Chen, Tzu-Ming Chi, "on the improvements of the particle swarm optimization algorithm," in *Advances in Engineering Software*, No. 41, pp. 229-239, 2010.

[13]J. C.Bansal, P. K. Singh, Mukesh Saraswat,Abhishek Verma, Shimpi Singh Jadon, Ajith Abraham ,''Inertia weight strategies in Particle swarm optimization,'' in *Proc. IEEE Int. Conf. Neural Netw.*, Perth, Australia, 1995, vol. 4, pp. 1942-1948.

[14]Particle Swarm Optimization and Differential Evolution Algorithms: Technical Analysis,Applications and Hybridization Perspectives

[15]Baskaran Anand, Indoria Aakash, Akshay, Varatharajan Varrun,Murali Krishna Reddy,Tejaswi Sathyasai, M.Nirmala Devi ,''Improvisation of Particle Swarm Optimization Algorithm, '' in*Int. Conf. Signal Processing and Integrated Networks (SPIN), India, 2014.*

[16]Ankunda R. Kiremire,''The Application of Pareto Principle in Software Engineering, '' *19th October, 2011.*

[17] Wikipedia.Paretoprinciple,[http://en.wikipedia.org/wiki/pareto principle](http://en.wikipedia.org/wiki/pareto_principle), Accessed March 2016.

[18]VirtualLibrary of Simulation Experiments:Test Functions and Datasets ,<http://www.sfu.ca/~ssurjano/>,Accessed March 2016.