**A modified decreasing inertia weight in Particle Swarm Optimization (PSO)**

Iqra new logo

*By:*

Muhammad Husnain

Reg No: 01727

*Supervised By:*

Dr. Hajira Jabeen

Assistant Professor

IQRA University Islamabad Campus

Thesis Submitted as partial fulfillment of requirement for the Degree of Master of Science in Computer Science (MSCS)

**DEPARTMENT OF COMPUTING AND TECHNOLOGY**

**IQRA UNIVERSITY ISLAMABAD CAMPUS**

October 2013

**Abstract**

Particle Swarm Optimization (PSO) algorithm has shown good performance in many optimization problems [[1](#Wen)]. PSO is a simple and one of fast evolutionary computing algorithm [[2](#Par)]. It has few operators as compared to other variants. This research work focuses on the inertia weights used in PSO to update the velocity of PSO. Inertia weight in PSO is used to enhance the searching ability of PSO algorithm. In this thesis a new modified decreasing inertia weight is proposed. Different inertia weights available in the literature are compared to assess the performance of these inertia weights and the proposed inertia weight. The experimental result show the proposed inertia weight enhances the performance of the PSO algorithm.

**ACKNOWLEDGEMENTS**

Up and above everything, I am grateful to almighty ALLAH, the beneficent, The Merciful, and His Prophet (Peace be upon him) who is forever a true torch of guidance for whole humanity. I am greatly obliged to “ALLAH” by whom grace, I have been able to complete this project successfully. I am especially indebted to my supervisor Dr. Hajira Jabeen, for giving me an initiative to this project. I am very thankful to Dr. Hajira Jabeen for her inspiring guidance, remarkable suggestions, constant encouragement, keen interest, constructive criticism, and friendly discussion enabled me to complete this thesis efficiently. Without her support and proper guidance, it would be almost impossible to accomplish this task successfully.

I am grateful to my loving parents for providing me all sort of moral and social support in life. Their prayers have enabled me to reach this stage.

Finally I again thank GOD for his uncountable blessings.

**DECLARATION**

I hereby declare that the work done in this research is my original piece of work performed under the supervision of Dr. Hajira Jabeen, Assistant Professor, Iqra University, Islamabad. Material from other authors, researchers has been properly acknowledged where-ever included in this report. And further I also certify that no part of this work, separate or as a whole has been presented for award of any other degree program.

Name: Muhammad Husnain

(MS (CS), 01727)

***Dedication***

***To hands,***

***Shivering and uplifted***

***Eyes heavy and thoughtful***

***Of my parents;***

***Hands ever praying for me***

***Eyes with dreams in of my bright tomorrow***

***These hands may never fall down.***

***These eyes may never go to asleep.***

***This Thesis is dedicated to my most respectful and honorable Parents and Teachers.***

**THESIS APPROVAL SHEET**

It is certify that Muhammad Husnain Student of MS (CS) Department of Computing & Technology, Student ID (01727) of IQRA University Islamabad, has submitted the final Thesis report on “**A modified decreasing inertia weight in Particle-Swarm Optimization (PSO)”** We have read the report and it fulfills the partial of Master of Science in Computer science degree.

**INTERNAL EXAMINER:**

**Name:**

Designation:

Organization: Iqra University Islamabad Campus

**Signature:** **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**EXTERNAL EXAMINER:**

**Name:**

Designation:

Organization:

**Signature:** **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**SUPERVISOR:**

**Name:**  Dr. Hajra Jabeen

Designation: Assistant Professor

Organization: Iqra University Islamabad Campus

**Signature:** **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Table of contents

[1. Chapter 1 - Introduction to function optimization 11](#_Toc370215853)

[1.1. Motivation 12](#_Toc370215854)

[1.2. Objective 12](#_Toc370215855)

[1.3. Contribution 13](#_Toc370215856)

[2. Chapter 2 - Introduction to Swarm Intelligence and Particle Swam Optimization (PSO) 14](#_Toc370215857)

[2.1. Swarm Intelligence 14](#_Toc370215858)

[2.2. Particle Swam Optimization (PSO) 14](#_Toc370215859)

[2.3. PSO Variants 16](#_Toc370215860)

[2.3.1. Initialization 16](#_Toc370215861)

[2.3.2. Constriction Coefficient 17](#_Toc370215862)

[2.3.3. Inertia Weight 18](#_Toc370215863)

[2.3.3.1. Constant Inertia weight 18](#_Toc370215864)

[2.3.3.2. Uniform Random Inertia Weight 19](#_Toc370215865)

[2.3.3.3. Adaptive Dynamic Weight Scheme 20](#_Toc370215866)

[2.3.3.4. Sigmoid Increasing and Decreasing Inertia weight 21](#_Toc370215867)

[2.3.3.5. Linear Decreasing Inertia Weight 22](#_Toc370215868)

[2.3.3.6. Chaotic and Chaotic Random Inertia weight 23](#_Toc370215869)

[2.3.3.7. Oscillating Inertia weight 24](#_Toc370215870)

[2.3.3.8. Simulated annealing Inertia weight 26](#_Toc370215871)

[2.3.3.9. Natural Exponent Inertia weight 27](#_Toc370215872)

[2.3.3.10. Logarithm Decreasing Inertia weight 28](#_Toc370215873)

[2.3.3.11. Exponent Decreasing inertia weight 29](#_Toc370215874)

[3. Chapter 3 - Literature Review 30](#_Toc370215875)

[4. Chapter 4 – Proposed inertia weight technique 36](#_Toc370215876)

[4.1.1. Proposed technique Inspiration 36](#_Toc370215877)

[4.1.2. Pseudo code for proposed technique 37](#_Toc370215878)

[4.1.3. Flowchart of Proposed technique 38](#_Toc370215879)

[5. Chapter 5 - Experimental Setting , Test Functions and Analysis 39](#_Toc370215880)

[5.1. PSO Inertia Weight Techniques used in experiments 39](#_Toc370215881)

[5.2. PSO Parameter settings 40](#_Toc370215882)

[5.3. Test Functions 41](#_Toc370215883)

[5.3.1. Sphere Function 41](#_Toc370215884)

[5.3.2. Griewangk’s function 42](#_Toc370215885)

[5.3.3. Rosenbrock’s Valley 43](#_Toc370215886)

[5.3.4. Rastrigin’s Function 44](#_Toc370215887)

[5.3.5. Ackley’s function 45](#_Toc370215888)

[5.3.6. Schwefel’s Function 46](#_Toc370215889)

[5.3.7. De Jong's function 47](#_Toc370215890)

[5.3.8. Axis parallel hyper-ellipsoid function 48](#_Toc370215891)

[5.4. Results 49](#_Toc370215892)

[5.5. Results Analysis 52](#_Toc370215893)

[5.6. Convergence Graphs 53](#_Toc370215894)

[6. Chapter 6 - Conclusion and Future work 57](#_Toc370215895)

[6.1.1. Conclusion 57](#_Toc370215896)

[6.1.2. Future Work 57](#_Toc370215897)

[Works Cited 57](#_Toc370215898)

List of Figures

[Figure 1: Graphical Overview of Sigmoid increasing and decreasing inertia weight [9] 22](#_Toc366441960)

[Figure 2: Graphical Overview of Chaotic and Chaotic Random Inertia weight [11] 24](#_Toc366441961)

[Figure 3: Flowchart of Global-Local Best Inertia weight 27](#_Toc366441962)

[Figure 4: Graphical overview of Natural Exponent Inertia weight [11] 29](#_Toc366441963)

[Figure 5: PSO with (a) Random population initialization (b) Opposition-based population initialization [4] 32](#_Toc366441964)

[Figure 6: Flowchart of proposed inertia weight technique 41](#_Toc366441965)

[Figure 7: A Graphical Overview of De Jong’s Function [32] 44](#_Toc366441966)

[Figure 8: Graphical overview of Griewangk’s function [32] 45](#_Toc366441967)

[Figure 9: Graphical Overview of Rosenbrock’s Valley [32] 46](#_Toc366441968)

[Figure 10: Graphical overview Rastrigin’s function [32] 47](#_Toc366441969)

[Figure 11: Graphical Overview of Ackley’s function [32] 48](#_Toc366441970)

[Figure 12: Graphical Overview of Schwefel’s Function [32] 49](#_Toc366441971)

[Figure 13: Graphical overview De Jong's function [32] 50](#_Toc366441972)

[Figure 14: Graphical overview Axis parallel hyper-ellipsoid function [32] 51](#_Toc366441973)

[Figure 15: Sphere function convergence graph 56](#_Toc366441974)

[Figure 16: Griewank Function convergence graph 57](#_Toc366441975)

[Figure 17: Rosenbrock Function convergence graph 58](#_Toc366441976)

[Figure 18: De Jong’s Function convergence graph 59](#_Toc366441977)

List of Tables

[Table 1. Parameters and values 43](#_Toc366441942)

[Table 2. Objective functions and their optimal values 43](#_Toc366441943)

[Table 3. Comparative analysis of inertia weight strategies with dimensions=10 and particles=30 52](#_Toc366441944)

[Table 4: Comparative analysis of inertia weight strategies with dimensions=20 and particles=30 53](#_Toc366441945)

[Table 5: Comparative analysis of inertia weight strategies with dimensions=30 and particles=30 54](#_Toc366441946)

1. **Chapter 1 - Introduction to function optimization**

Optimization means to find the optimal element from available pool of elements. Function optimization is used to minimize or maximize the function subject to some constraints. Some ingredients of optimization are listed as

* **Cost function**: which represents the quantity to be optimized that is the quantity to be minimized or maximized.
* **Decision variables**: A set of unknowns or variables which affect the value of the objective function.
* **Constraints**: A set of constraints that restrict the values that can be assigned to the unknowns. Most problems define at least a set of boundary constraints which define the domain of values for each variable.

Optimization problems are classified by

* The number of variables that influence the objective function

A problem which based on a single variable for optimization is referred as uni-variable, while with more variables is referred as multivariable.

* The type of variables

Different types of problem are like

1. Continues problem contain continues valued variables
2. Discrete problems consist of integer valued variables
3. Mixed both continues and discrete, based on both continues and integer valued variables
4. Combinatorial optimization problem solutions are permutations of integer valued variables.

* The degree of nonlinearity of the objective function

Linear problem used linear objective function, quadratic used quadratic objective function while non linear problem used non linear objective function.

* The constraints used

A problem is known as unconstrained problem if it has only boundary constraint. If a problem has additional equality and inequality constraints called constrained problem

* The number of optima

Some problem has only one optima and some has more than one.

* The number of optimization criteria

Two types of problems are uni-objective and multi-objective problems.

In evolutionary field different algorithms are compared using a large test set particularly in case of function optimization. The objective of an optimization method is to allocate values from the allowed domain to the unknowns such that the objective function is optimized and all constraints are contented.

Optimization algorithms are search methods where the objective is to find a solution to an optimization problem such that a specified quantity is optimized, possibly subject to a set of constraints. Solutions originated by optimization algorithms are classified by the quality of the solution. The major types of solutions are referred to as local optima or global optima.

An optimization algorithm searches for an optimum solution by iteratively transforming a current candidate solution into a new, expectantly improved solution. Optimization methods can be divided into two major classes based on the sort of solution that is located. Local search algorithms use only local information of the search space surrounding the current solution to produce a new solution. A global search algorithm uses more information about the search space to locate a global optimum. Optimization algorithms are further classified into deterministic and stochastic methods. Stochastic methods use random elements to convert one candidate solution into a new solution. Deterministic methods do not employ random elements.

* 1. **Motivation**

Function optimization is a generic problem which can be easily resolved to any real life optimization problem like travelling sales man problem (TSP) or time table scheduling problem etc. As Particle swarm optimization (PSO) is theoretically proved for optimization problem therefore PSO is used in this thesis.

* 1. **Objective**

Inertia weight is used to control the step size in velocity updation of PSO. Inertia weight enhances global search ability of PSO algorithm. Different researchers have used different inertia weight to update the equation of velocity. The performance of one inertia weights may be different from other inertia weights. The objective of research is to perform a comparative analysis of different inertia weight in PSO. The comparison will be performed by using different optimization functions taken from the literature.

* 1. **Contribution**

Contribution in this research is a modified decreasing inertia weight in Particle Swarm Optimization (PSO) is introduced. Various inertia weights used by other researchers are also included in the research for function optimization problems. Research results will help to choose the best performing inertia weight in PSO for any optimization application.

**Thesis outline**

In chapter 2 different variants of PSO are discussed in detail. Chapter 3 contains literature review. Chapter 4 is about proposed inertia weight technique. Chapter 5 contains experimental settings, test functions and results of test functions with different inertia weight techniques. Chapter 6 is about the conclusion and future work.

1. **Chapter 2** **- Introduction to Swarm Intelligence and Particle Swam Optimization (PSO)**
   1. **Swarm Intelligence**

Swarm intelligence (SI) is the intelligent behavior of non intelligent species, like ants (going toward search of food) or birds (during flying). SI system is made up by the simple independent population entities interacting with their environments and each other. There is no central control dictating the performance of these independent entities.

To explore discrete problem solving without having a central control structure the artificial intelligence used is called SI. Real life swarm intelligence can be observed in ant colonies, bacterial growth, bird flocks, animal herds and fish schooling.

One popular model of swarm behavior is Ant colony model. Ant colony behavior is one of the popular models of swarm behavior. Through swarm intelligence ants can determine the shortest path to source of food. Ant colony optimization be used to solve traveling salesman problem, Scheduling problem, Vehicle routing problem and many other problems.

Particle swarm optimization is a type of swarm intelligence inspired by bird flocking and fish schools. This type of swarm intelligence is used in practical applications such as in artificial neural networks and in grammatical evolution models.

* 1. **Particle Swam Optimization (PSO)**

PSO is a population based optimization method purposed by Kennedy and Eberhart [[2](#Par)]. The algorithm simulates the behavior of bird flock flying together in multi dimensional space in search of some optimum place, adjusting their movements and distances for better search. PSO is very similar to evolutionary computation such as Genetic algorithm (GA). The swarms are randomly initialized and then search for an optimum by updating generations.

PSO is a combination of two approaches, one is cognition model that is based on self expression and the other is a social model, which incorporates the expressions of neighbors. The algorithm mimics a particle flying in the search space and moving towards the global optimum. A particle in PSO can be defined as where i=1, 2, 3…. D and a, b , D is for dimensions and R is for real numbers . All the particles are initialized with random positions and with random velocities [[2](#Par)], then particles move towards the new position based on their own experience and with neighborhood experience. Each particle in PSO maintains two important positions called pbest and where the particle own best position is and t is the global best position among all the particles. The equations used to update a particle’s velocity and position, are the following

-----------------------------------------------------------2.1

---------------------------------------------------------------------------------------------2.2

Where d=1, 2, 3….n dimensions, i=1, 2, 3……s is particle index, c1 and c2 are constants

* 1. **PSO Variants**

As originally PSO proposed by R. Eberhart and J. Kennedy [[2](#Par)] in 1995, Researchers are trying to make this technique as best as possible, lot of PSO variants have been proposed by researchers with respect to different parameters and operators, the detail of some of them is given as.

* + 1. **Initialization**

Initialization of population plays an important role in the evolutionary and swarm based algorithms. In case of bad initialization, the algorithm may search in unwanted areas and may be unable to search for the optimal solution.

Nguyen et al [[3](#Ini07)] inspect the some randomized low discrepancy sequence to initialize the swarm to increase the performance of PSO. They used three low discrepancy sequence Halton, Faur and Sobol. Halton sequence is actually the extension of van der Corput. Ven Der Corput sequence is one dimensional in order to cover search space in n dimension and Halton is defined as one of the extension of Vender Corput sequence. They compare their all three new variants with global best fitness of PSO in which swarms are initialized with pseudo random number. It is observed from the results that S-PSO is dominated among all the four version of the PSO. In case of small search space PSO initialized with Faur sequence can perform well while in case of high dimensions Halton performance might be fine. Six Benchmark functions are used to evaluate the performance of new three version of PSO

Pant et al [[4](#MPa)] explore the effect of initializing swarm with the vender Corput sequence which is a low discrepancy sequence to solve the global optimization problem in large dimension search space. They named the proposed algorithm as VC\_PSO. The author claim that PSO performance is very well for problems having low dimensions but as the dimensions evolve the performance deteriorates, this problem become more severe in case of multimodal functions. The author says that one of the reasons for this poor performance may be random initialization of the swarm therefore they proposed a PSO technique which initializes the swarm with low discrepancy random number to overcome this problem. They compare their algorithm with PSO using Sobol random sequence which is dominated by Halton and Faur sequence. Vender Corput is a low discrepancy sequence over the unit interval proposed by a Dutch mathematician in 1935, which is defined by the radical inverse function. They used the linear decreasing inertia weight from .9 to .4 with c1=c2=2.0.

Jabeen et al [[5](#Opp1)]proposed opposition based initialization which calculates opposition of randomly initialized population and selects better among random and opposition as initial population. This population is provided as an input for traditional PSO algorithm. The proposed modification has been applied on several benchmark functions and found successful.

Chang et al [[6](#Cha09)] proposed an enhanced version of opposition based PSO called quasi-oppositional comprehensive learning PSO (QCLPSO). Instead of calculating traditional opposite of a point, the proposed modification calculates Qausi opposite particle, which is generated from the interval between median and opposite position of the particle. According to authors the Qausi opposite particles have higher chances of being closer to global optima then opposite particle calculated without apriori information.

* + 1. **Constriction Coefficient**

This approach to balance the exploration-exploitation trade off has been proposed by Clerc [38], which uses a new parameter ‘χ’ called the constriction factor. The velocity update scheme proposed by Clerc can be expressed for the dth dimension of ith particle as:

--------------------------2.3

Where χ=

With C1+C2, ,

Constriction coefficient results in the quick convergence of the particles over time. That is the amplitude of a particle’s oscillations decreases as it focuses on the local and neighborhood previous best points. Though the particle converges to a point over time, the constriction coefficient also prevents collapse if the right social conditions are in place. The particle will oscillate around the weighted mean of and, if the previous best position and the neighborhood best position are near each other the particle will perform a local search.

* + 1. **Inertia Weight**

The inertia weight is a scaling factor associated with the velocity during the previous time step resulting in a new velocity update equation introduce by Shi. Inertia Weight is used to control the exploration and exploitation abilities of the swarm. Large value of inertia weight promotes exploration while small value promotes local exploitation.

* + - 1. **Constant Inertia weight**

Shi and Eberhart [[7](#YSh98)] introduced the concept the inertia weight and that inertia weight value was constant. Researchers used fixed value of inertia weight and that inertia weight is called as constant inertia weight. Including inertia weight in the equation means how much amount of previous velocity is used in current iteration. Their stated that when there is case of global search then large inertia weight is used and when local search then small value of inertia weight is used.

Velocity update equation used is as under

--------------------------------------------------2.4

--------------------------------------------------------------------------------------------2.5

When inertia weight was included, then equation becomes as under

--------------------------------------------2.6

“w” used indicates the inertia weight. For velocity control purpose, inertia weight is used.

* + - 1. **Uniform Random Inertia Weight**

Zhang et al [[8](#Qin07)]set the inertia weight as uniformly random number between 0 and 1. Author claims that it is more capable to escape from local minima. According to author’s proposed method for inertia weight, can overcome two problem of linearly decreasing inertia weight.

* We can overcome the problem of linearly inertia weight dependency on maximum iteration.
* Another is avoiding the lack of local search ability at early of run and global search ability at the end of run.

They test their method on three benchmark functions with different dimensions using different number of generations. The result of new proposed inertia weight found best.

Formula used is as under

--------------------2.7

--------------------------------------------------------------------------------------------2.8

The first equation is used velocity update and then second one is for position update.

* + - 1. **Adaptive Dynamic Weight Scheme**

Shu-Kai [[9](#SKa)] proposed PSO using an adaptive dynamic weight scheme. They propose a novel nonlinear function amendable inertia weight adaptation with an active method for improving the performance of PSO algorithms.

The aim was the determination of the inertia weight through a nonlinear function at each time step. The nonlinear function is given by

………………………………………………………….………………………….2.9

Where d is the decrease rate from 1.0 to 0.1 and r is dynamic adaption rule depending on the following rule. For minimization case it follows.

If then ………………………...……………..…….. 2.10

If then ……… ………… …… ………………….. 2.11

Where and represent the global best position at current and previous time step respectively. Author [[10](#CSF07)]claims that this method desires to make particle take off quickly towards the optimal solution and then perform local enhancement around the neighborhood of finest solution by decreasing inertia weight. They test their technique using different benchmark function and find best results.

* + - 1. **Sigmoid Increasing and Decreasing Inertia weight**

Auther RF Malik et al [[11](#RFM07)] used sigmoid function which is either increasing or decreasing to improve the performance of PSO. The properties of sigmoid function are that either it is implemented in increasing or decreasing way.

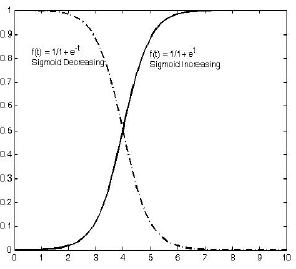


Figure 1: Graphical Overview of Sigmoid increasing and decreasing inertia weight [9]

The basic of sigmoid function is given as:

-----------------------------------------------------------------------------------------------2.12

(Sigmoid increasing)-----------------------------------------------2.13

(Sigmoid decreasing)-----------------------------------------------2.14

----------------------------------------------------------------------------------2.15

w start and w end are inertia weights at start and end. u is constant to adjust sharpness of function. gen are maximum number of generations to run. The Inertia weight is implemented using sigmoid curve. In sigmoid decreasing, large inertia weight is maintained for global search and then increasing for local search. In sigmoid increasing, small inertia weight for local search.

* + - 1. **Linear Decreasing Inertia Weight**

In linearly decreasing inertia weight, large value (0.9) linearly decreased to small value (0.4). Formula used for linearly decreasing inertia weight is following.

------------------------------------ 2.16

Where is 0.9 and is 0.4 is the maximum number of iterations, is the iteration during which inertia weight is calculated.

The velocity of linear decreasing inertia weight is as under

-----------2.17

This inertia weight was proposed by Shi and Ebarhat [[12](#RCE01)]. Overall performance of PSO is improved by this type of inertia weight because it balances the global and local search abilities of swarms.

**Pseudo code is as under**

* Begin
* Randomly initialize particle swarm
* While (termination condition)
* Evaluate fitness of swarm
* For n=1 to number of particles
  + *Find pbest*
  + *Find gbest*
* For d=1 to number of dimensions of particle
* Update position of particles
* next d
* next n
* Update inertia weight value
* Next generation until stopping criteria is met
  + - 1. **Chaotic and Chaotic Random Inertia weight**

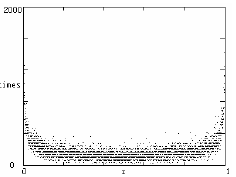
This strategy was introduced by “chaotic optimization mechanism” in PSO [[13](#Yon06)] . When chaotic inertia weight performance is compared with random inertia weight, then it performs outstanding results. In this case chaotic mapping is used to set coefficient of inertia weight. Logistic mapping is an alternative method to do it.

Figure 2: Graphical Overview of Chaotic and Chaotic Random Inertia weight [11]

The figure above is the distributing figure of logistic mapping.

Chaotic inertia weight has different strategies

* Chaotic descending inertia weight
* Chaotic random inertia weight

In Chaotic random inertia weight, random number z is selected between interval 0 and 1. Then logistic mapping is made.

Following formula is used

------------------------------------------------------------------------2.18

Here w1 and w2 are initial and last values of inertia weights. MAXiter is maximum number of iterations and iter is current iteration

In Chaotic random inertia weight, a random z number is selected between interval 0 and 1. A random number Rand( ) is selected between the interval 0 and 1. Logistic mapping is made.

Following formula is used.

-----------------------------------------------------------------------------2.19

These two strategies have chaotic characteristic and they have varying trend.

* + - 1. **Oscillating Inertia weight**

In this kind of inertia weight strategy, a wave of global search is following by a wave of local search [[14](#Kyr09)] . We can also say that exploration is followed by exploitation. This strategy is repeated until the optimization process is completed. In this strategy, the swarm periodically moves from global to local search. This kind of behavior is called Temporal Behaviour that is implemented by means of inertia weight function w(t).

In sinusoidal function with constant amplitude, the formula is given as

---------------------------------------------------2.20

Is the inertia weight at start and is the inertia weight at the end

The number of iterations competed for inertia weight completed is give by the following formula.

-----------------------------------------------------------------------------------------------2.21

Here S is the number of iterations for which inertia weight is allowed to oscillate. K is a parameter that controls the frequency of oscillation.

In sinusoidal function with linear decreasing amplitude, the swarm transits between global and local search. The value of inertia weight now oscillates between Wmin and Wmax as shown by the given equation

-----------------------------------------------2.22

By using the step function, same features are used as in sinusoidal function. In this case, exploration and exploitations are discretely modulated according to following equation

---------------------------------------------------------------------------2.23

When Oscillating inertia weight is tested using some benchmark functions, then it is observed then it performs well in some cases. In some cases it outperforms in term of consistency and speed of convergence.

**Global-Local Best Inertia weight**

This kind of inertia weight is neither set constant nor decreasing by time varying factor. The value of this inertia weight is basically depends upon global best (gbest) and personal best (pbest).

The formula for Global-Local best inertia weight is given as below

Inertia weight w = --------------------------------------------------------------2.24

Here is particle local best value and is best value particle in the swarm.

Read in data and define constraints

Initialize the swarm;

1. Randomize each particle
2. Randomize velocity of each particle
3. Select type of PSO from table
4. Run optimal power flow
5. Initialize each Pbest equal to current position of each particle

Update iteration count

Update velocity and position of particle

Get new particle position

Update Pbest if new position is better than Pbest

Update Gbest if new position is better than Gbest

Repeat for each particle

Stopping criteria satisfied

Gbest is the optimal solution

Stop

Figure 3: Flowchart of Global-Local Best Inertia weight

When there is need to solve any high dimensional optimal control problem, with high convergence rate and accuracy, then this kind of inertia weight is used as it concerns about global best and personal best.

* + - 1. **Simulated annealing Inertia weight**

This technique is also called as “Random Search Technique”. This technique is based on Monte Carlo simulation. This method is used in solving combinational optimization problems. This technique was introduced by Kirkpatrick, Gelatt and Vecchi [[15](#WAl061)]. The regions near local minima are removed through simulated annealing. This can be used as global maximization like bouncing ball. The generating distribution is used to explore states or valleys.

The acceptance distribution is depends upon the current state and last solved explored state. The accepted distribution also depends upon the temperature.

Simulated annealing better helps particle swarm optimization for good solution. It helps to better control the convergence of PSO.

Formula is given as below

--------------------------------------------------2.25

Where the new inertia is weight and is the previous inertia weight. N indicates the total number of particles in the swarm and ns are the total steps taken by the simulated annealing technique.

When the new calculated value is less than previous calculated value then inertia weight is updated by the new calculated value because it is less than previous one.

For acceptance probability, Gaussian function is used. The formula for Gaussian function is as

Under

--------------------------------------------------------2.26

Where P(t) is the probability of acceptance. Tem is coefficient of temperature that is changed at each iteration. is fitness at current calculation and is the fitness at previous calculation.

Temperature change formula is given as

--------------------------------------------------------------------------2.27

During experimental results, it has been observed that convergence rate of PSO using SA inertia weight is better than standard PSO.

* + - 1. **Natural Exponent Inertia weight**

There are two natural exponent strategies inertia weight strategies those are e-based [[16](#Gui06)] . First one is represented in equation form as under

-------------------------------------------------------2.28

adopting this strategy is called e-pso1.

-------------------------------------------------------2.29

The 2nd natural exponent strategy is as under

PSO adopting this strategy is called e-pso2.

Here is the inertia weight at start and is the inertia weight at the end.

The comparison of three decreasing strategies by graph is as under

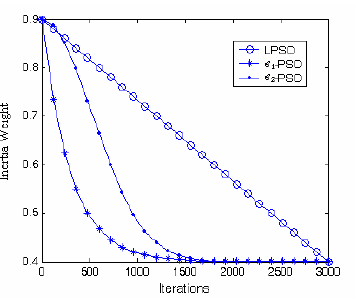


Figure 4: Graphical overview of Natural Exponent Inertia weight [11]

When experiments are done on different functions like Sphere, Rosenbrock, Griewank, Rastrigin using linear decreasing and two natural exponent inertia weight strategies, then it has been observed that natural exponent strategies perform well and converge faster as compared to linear decreasing strategy during the early the early stage of convergence. These strategies give better results for most of the optimization problem.

* + - 1. **Logarithm Decreasing Inertia weight**

This inertia weight strategy was basically implemented with Chaos mutation operator [[17](#Yue08)]. This technique can acquire better performance of PSO regarding convergence. The purpose of Chaos mutation operator is to jump out from local minima and in order to escape from premature convergence.

The formula for logarithm decreasing inertia weight is given as

-----------------------------------------------------2.30

Is the inertia weight at start and is the inertia weight at the end.

The “a” used in this equation is constant and is used for the adjustment of evolutionary speed.

Chaos is considered as nonlinear dynamic function. This is applied on stochastic optimization.

This function has many advantages in the local optimization.

Logarithm decreasing inertia weight and Chaos mutation operator work together to get the optimum results of the algorithm.

To judge at what time premature convergence occurs, two strategies were introduced. This algorithm is good in convergence as well as in accuracy.

* + - 1. **Exponent Decreasing inertia weight**

The exponent decreasing inertia weight is introduced by Hui [[18](#Hui09)] and its equation is

-------------------------------------------------------------2.31

Where denotes the max iteration, t denotes the tth iteration. denotes the original inertia weight, denotes the inertia weight value when the algorithm process run the max iterations, and is a factor to control w between .

In order to beat the early convergence Hui proposed a variant of PSO with exponent decreasing inertia weight and stochastic mutation. Exponent decreasing inertia weight describe in equation Mutation probability pm is defined as

= ------------------------------------------------------2.32

Where >0, is the fitness of current global particle, Fm is the theory optimum value of the optimal problem, is defined as

---------------------------------------------------------------------------------2.33

Where, f is factor of returning. It can be limit value of f is defined as

----------------------------------------------2.34

Gbest is mutated as following

----------------------------------------2.35

Where is random variable and it denotes the standard normal distribution.

1. **Chapter 3 - Literature Review**

Jabeen et al [[5](#Opp1)] introduced new algorithm for initialization of population, called O-PSO. Here performance of O-PSO is compared with existing PSO on several benchmark functions.

Initialization of PSO plays a key role. Random selection can result in exploiting the search area. But random selection from opposition population lowers the chances of exploitation.

In this technique, opposites of initial population are selected and are chosen for evaluation. O-PSO increases the chance of convergence speed.

Initialize Random Swarm P (n)

PSO

Calculate opposite of Swarm P (n)

Select n fittest individuals from Swarm P (n)

Initialize Random Swarm P (n)

PSO

(a)

Figure 5: PSO with (a) Random population initialization and (b) Opposition-based population initialization [4]

In this method, population and their opposites are selected as input. The fitness of both are tested and only fitter ones are selected.

For experimental setting, four benchmark functions are used.

When comparison of PSO done with O-PSO, then it has been observed that O-PSO performs well in three functions out of four.

Ozcan, M Yllmaz [[19](#EOz07)] reviewed five previous PSO algorithms and has proposed new algorithm named as CPSO. The proposed algorithm enhances the capabilities of PSO. The search is made by using the random walk and hill climbing components.

In the proposed algorithm, the main swarm is divided into sub-swarms of size “n” according to their geographical positions.

The process is started from the first particle, and the each particle which doesn’t belong to sub-swarm is detected. The re-arrangement of particles is done according to their geographical current positions. So communication and information diffusion took place between the particles. The particle can change value according to neighborhood local best value.

Moving toward better neighborhood position is named as hill climbing. The proposed algorithm performs well with low maximum velocity. The proposed algorithm is very successful for locating global optima as compared to local optima.

In the proposed algorithm, the cost factor is reduced by limiting the computations at pre-defined frequency.

Proper Initialization of PSO can better helps to explore the search space. K. E. Parsopoulos, M. N. Vrahatis [[20](#KEP02)] proposed method called Linear Simplex Method. This method work better especially for noise functions. The proposed method takes series of steps where vertex of function with highest function value is moved.

The swarm is informed that what are the better regions and this information is passed as vertex of function NSM in each step.

The proposed algorithm provides PSO faster and better regions of the search space. The proposed algorithm is computationally expensive and NSM and PSO are easily implemented.

Chang Zhang, Zhiwei Ni et al [[6](#Cha09)], introduced the improved version of opposition based PSO. This method is called Quasi Oppositional Learning PSO for the initialization of population. This is generated from opposite position of particle and median.

There is mathematical proof that Quasi-Opposite particles are very close to the solution as compared to opposite particles.

The proposed CLPSO and advantages as compared to OCLPSO, listed below

* Exemplars diversity: Instead of using all pbest as exemplars, all particles and their quasi opposites are taken as exemplars, so the more search space is available and their diversity is increased.
* Computing time: As the quasi particles are close to the solution, so computing time will be reduced.
* Exploration: As the search space is increased due to quasi opposite particles, so particles will explore more search space for locating the best possible solution.
* Premature convergence: The proposed algorithm discourages the premature convergence. When all the swarms come into local minima, then through quasi opposite phenomena, they pull out from that situation.

The proposed algorithm is best for multimodal problems.

M. G. H. Omran and S.al-Sharhan et al [[21](#Usi)] used an opposition based learning to improve the performance of PSO. In every iteration, the particle with lowest strength of fitness is replaced by it opposite. The speed and individual experience of the anti-particle are reset. After that a global best solution is updated. They have not introduced any new parameter to PSO. The only modification is the use of opposition based learning to enhance the performance of PSO.

When the dimensions of PSO are increased, the performance of the algorithm deteriorates. Andress P.Engelbrecht Frans van den Bergh [[22](#Fra041)] provided solution of this problem by dividing the search region into lower subspaces. The proposed algorithm is very robust even dealing with multimodal rotating functions. When there is case of unimodal function, then standard PSO and CPSO perform well in unrotated case.

When the proposed algorithm was tested against traditional PSO on several benchmark functions, the performance of the proposed algorithm was good.

Anthony Carlisle, Gerry Dozier [[23](#Ant09)] provided new technique. In this technique, each particle is given option to reset its record as the environment changes. There are two methods for initiating this process.

* Iteration based, when specific iterations are done, and then they reset their record. This method is also called iteration count basis. This can be done on regular frequency.
* When magnitude of the environment is changed, they are triggered to reset their value. The particles are triggered when they achieve specific distance from the original position.

Following parameters are recorded in the given experiments

* Reliability in term of solution found
* Efficiency in term of average iteration per solution found
* Median iterations per run

Another variant of PSO called comprehensive learning is introduced by [[24](#JJL06)]. In this strategy, each particle history is checked and is update according to its history.

A random number is generated. If this larger than PC then it will learn from own otherwise it will learn from neighbor “” particle. In order to minimize the time wasted on the poor directions, the particle learns from exemplars.

The proposed technique has maximum space area to fly. When this proposed technique is tested then it gave optimum results on multimodal functions when it is compared with other PSO variants.

In new proposed PSO by E Ozcan, M Yllmaz [[19](#EOz07)] called CPSO, search is performed using random walk and hill climbing component. In the proposed algorithm, the main swarm is divided into sub-swarms according to size “n” according to the geographical position. The swarms those don’t belong to sub-swarms are detected. The sub-swarms are re-arranged. This method provides a type of communication and diffusion of information between the particles and local best value is changed with in neighborhood. Hill climbing is defined as “Moving to better candidate position”. In local global optima, the proposed algorithm shows better results. In locating multiple global optima, the performance of proposed algorithm is good.

Basically PSO was proposed for the optimization of static environments. But there are some dynamic problems [[25](#SKi08)]. Dynamic problem is defined as the characteristics of global optimum changes with time. Author has checked the performance of PSO on multimodal functions and non-stationary environments.

A technique named as functional global best formation (FGBF) has been introduced. This technique is used to avoid the pre-mature convergence by providing significance diversity. Another technique called MDPSO has been also introduced. This basically allows inter-dimensional passes with dedicated PSO process.

The above both techniques find and track the optimum dimensions where global peaks found. Both technique upgrade particle structure. These are modular techniques and independent of each other.

Niche PSO was proposed by [[26](#ANi)] and is used to locate multiple optimization problems for multimodal problems. PSO using lbest and gbest can be used to for unimodal problem but not good in multimodal problems because of their social interaction of pbest and gbest. NichePSO was first used for solving multimodal problems [[26](#ANi)]. Here there is no sharing of information. In parallel niching several nichies are identified and maintained simultaneously. In sequential niching, multiple solutions are found by applying niching to problem space. A Guaranteed Convergence PSO is applied in optimization of subswarms.

Low discrepancy sequences were introduced by [[3](#Ini07)] and have been used with standard PSO. These sequences are as under. (i) Randomized sobol sequence (ii) Randomized halton sequence (iii) Randomized faure sequence. When PSO with uniform initialization was compared with PSO with low discrepancy sequences then proposed algorithm was best and had results with biggest success rate. It is worth replacing uniform sampling of initial population using the randomized low discrepancy sequences.

In hybrid PSO, the results of traditional velocity and position update are combined with idea of breeding and subpopulation. This idea was introduced by Morten Løvbjerg, Thomas Kiel Rasmussen, Thiemo Krink [[27](#Mor01)]. As in traditional PSO, the particles are moved in the search space. In sub-population idea, the gene-flow is restricted to evade sub-optional convergence. Particles are divided into sub-population and each sub-population has its own local best unique optimum. Here standard genetic algorithm was introduced. When the proposed technique compared with uni-modal and multi-modal problems, then it outperformed with uni-modal and performed better with multi-modal.

Dynamic sociometry in PSO was introduced by M Richards, D Ventura [[28](#MRi03)] .Performance of PSO is greatly affected by the size and sociometry of the swarm. As the basic function of PSO is to converge all the particles to optimum of the function. In the initial stage, all the particles have given the same velocity and later on velocity is adjusted to move it to its best position. In dynamic sociometry, the wide survey of the space is made and regions looking more promising are identified. Other sociometries are ring and star. Six benchmark functions are taken and experiments are done using dynamic sociometery and compared with ring and star sociometry. The results show that dynamic sociometry proved to be effective in some situations.

Arlindo Silva, Ana Paula Neves F. da Silva et al [[29](#Arl02)] introduced a predator-prey approach to function optimization In this PSO approach of PSO, new particles called predators are introduced. This approach is used for balancing between the exploitation and exploration by introducing the second population called predators. These have different dynamic behavior. The proposed algorithm creates diversity in the swarm during the time; the algorithm runs and avoids the particles from convergence. The influence of predator on individual is controlled by “fear” probability. When the proposed algorithm is considered in uni-model , then its results are not so good but it performs better form multi-model applications in optimization four bench mark functions.

Particle swarm is treated as “multi-agent system” and every particle is treated as agent [[30](#Wei09)]. The cause is that track the solution space simultaneously and to search global solution themselves.

By this improved method, the groups of comparing experiments have been done on four benchmark functions. Emotions are added to existed PSO algorithm and standard PSO and proposed PSO is compared. Emotions bring competition on PSO. Experimental results show that proposed algorithm has good affect on PSO.

A new technique “velocity clamping” is used in order to avoid pre-mature convergence, control velocity and speed of particles [[31](#Opp)]. If the velocity of particles exceeds the maximum allowed velocity, then it is set to maximum velocity. If the value of velocity is large, then it goes for exploration and step size is large and if value is small then it goes to exploitation and step size is small.

When the proposed technique is tested on four benchmark functions, then it performs better for uni-model and muti-model functions. However some functions of proposed PSO could not find the optimum results.

Srinivas Pasupuleti , Roberto Battiti [[32](#Sri06)] proposed a technique called Gregarious Particle Swarm Optimization (G-PSO).The proposed PSO explores the search space by searching promising regions rapidly in the search space. When in the second re-initializing phase, particles don’t lose their global exploration. As the search continues so it helps in finding the best optima. When the proposed algorithm was tested with standard PSO, then it shows outstanding performance. Here dynamic initialization is used to explore and exploit the search space. The propose PSO has high convergence rate and least error rate when we compare it with standard PSO.

1. **Chapter 4 – Proposed inertia weight technique**

A modified decreasing scheme of inertia weight technique is developed so that convergence performance of PSO can be improved. Inertia weight starts with maximum value in initial iterations and gradually decreases.

Following equation is used for proposed technique

0.0002 ---------------------------------------------------------------------------------4.1

is the maximum limit of inertia weight that is set to 1

is the current iteration

current iteration is multiplied with a small constant number 0.0002 (about 0.02% of iterations)

when iteration number increases, the value of inertia weight decreases.

From experimental results, it has been seen that proposed technique shows best results as compared to others

* + 1. **Proposed technique Inspiration**

A modified PSO, proposed by Shi and Eberhart, not only introduced the inertia weight but they also gave the idea that non linear decreasing functions can also be used to test the better performance of the PSO. This idea gave motivation to researchers. In order to improve the performance of PSO, a new inertia weight technique has been proposed. All the parameters used in this technique are same as the standard PSO except the inertia weight in order to compare its performance with other inertia weight techniques.

As the conventional PSO used constant inertia weight, the proposed inertia weight is not a constant value but it decreases gradually as the number of iterations increases. As inertia weight starts from large value, so it goes to exploration and then ended at small value and goes to exploitation.

The proposed technique has been tested on 8 benchmark functions and compared its performance with other inertia weights. Experimental results show that the proposed technique has best success rate for finding the optimal value as compared to other inertia weight techniques. The computational comparisons show that the proposed technique exhibits a noticeable advantage.

* + 1. **Pseudo code for proposed technique**

Initialize the population randomly

While (termination condition false)

{

Loop

Calculate fitness

If fitness value is better from the best fitness value () in history then

Update with the new

End loop

Select the particle with the best fitness value from all particles as

While maximum iterations or minimum error criteria is not attained

{

For each particle

Calculate particle velocity by equation (2.1)

Update particle position according to equation (2.2)

Update velocity using proposed technique given in equation (4.1)

Next

}

}

* + 1. **Flowchart of Proposed technique**

The flowchart of proposed inertia weight technique is shown in figure

Read in data and define constraints

Initialize the swarm;

1. Randomize each particle
2. Randomize velocity of each particle
3. Select type of PSO from table
4. Run optimal power flow
5. Initialize each Pbest equal to current position of each particle

Update iteration count

Update velocity and position of particle

Get new particle position

Update velocity using proposed inertia weight technique

Repeat for each particle

Stopping criteria satisfied

Stop

Gbest is the optimal solution

Update velocity using proposed w technique

Figure 6: Flowchart of proposed inertia weight technique

1. **Chapter 5 - Experimental Setting , Test Functions and Analysis**

In this chapter the results of fifteen PSO inertia weight techniques are compared using some standard Benchmark functions. Following paragraphs will give the description about the comparison of techniques. The analysis of these results will also discuss in this chapter.

* 1. **PSO Inertia Weight Techniques used in experiments**

1. Constant inertia weight =0.7
2. Random inertia weight =0.5+(rand()/2)
3. Adaptive Inertia weight
4. Sigmoid Increasing Inertia weight
5. Sigmoid Decreasing Inertia weight
6. Linear Decreasing Inertia weight
7. The Chaotic Inertia weight
8. Chaotic Random Inertia weight
9. Oscillating Inertia weight
10. Global Local Best Inertia weight
11. Simulated Annealing Inertia weight
12. Natural Exponent Inertia weight (e1-pso)
13. Natural Exponent Inertia weight (e2-pso)
14. Logrithem Decreasing Inertia weight
15. Exponent Decreasing Inertia weight
16. Proposed Modified Decreasing Inertia weight 0.0002

**PSO Parameter settings**

Table 1. Parameters and values

|  |  |
| --- | --- |
| Parameter | value |
| Dimensions | 10 |
| 20 |
| 30 |
| Iterations | 5000 |
| Population size | 30 |
| Number of PSO Runs | 30 |

Table 2. Objective functions and their optimal values

|  |  |  |  |
| --- | --- | --- | --- |
| Function Name | Objective Function | Search space | Optimal value |
| Sphere |  |  | 0 |
| Grienwank |  |  | 0 |
| Rosenbrock |  |  | 0 |
| Rastrigin |  |  | 0 |
| Ackley |  |  | 0 |
| Schwefel |  |  | 0 |
| De Jong’s |  |  | 0 |
| Axis parallel hyper-ellipsoid | ) |  | 0 |

* 1. **Test Functions**
     1. **Sphere Function**

Sphere is the simplest test function. It is continues, convex and uni-model function.Local optimizers are easily identified using this function. In various experiments when optimal values and standard deviations are observed, then it generates optimum results for various iterations. This function is also called as single model function means they have single optimum value for testing global convergence of system. Any Optimization technique can be easily applied. General definition of the function is given below.

-------------------------------------------------------------------------------------------------5.1

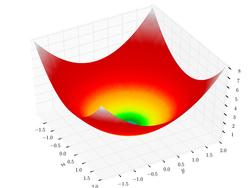
****

Figure 7: A Graphical Overview of De Jong’s Function [[33](#HPo)]

* + 1. **Griewangk’s function**

This function is similar to Rastrigin’s function. It has many widespread local minima, while the locations of minima are regularly distributed. The Griewangk’s function becomes easier as the dimensions increase. When different procedures are adopted for finding local minima, then this function is used for to test their performance.

------------------------------5.2

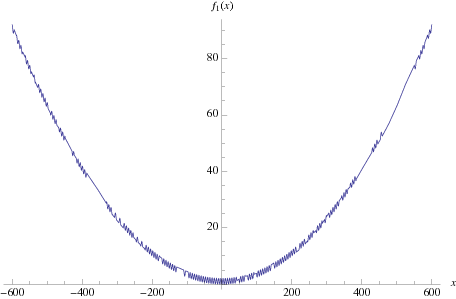


Figure 8: Graphical overview of Griewangk’s function [[33](#HPo)]

* + 1. **Rosenbrock’s Valley**

This function is known as banana function or the second function of de jong. It is non convex function introduced by Rosenbrock in 1960. The global minimum is inside a long, narrow, parabolic shaped flat valley. This function is used to test the performance of optimization problem. Without building local approximation models, this function is efficiently optimized. The function definition is ----------------------5.3

It has a global minimum at (x, y)=(1, 1)where f(x, y)=0

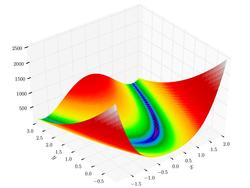
Sometimes, position of second coefficient is given but it does not affect the position of global minimum.

Figure 9: Graphical Overview of Rosenbrock’s Valley [[33](#HPo)]

* + 1. **Rastrigin’s Function**

This function is based on the function of de jong with addition of cosine modulation to produce many local minima and thus it is a multi-model function. This is non-convex model and used to test the problem for optimization. The locations of minima are distributed on regular basis. To test genetic algorithm, this function is mostly used. Because this function has more local minima and more search space, that’s why this function is difficult problem.

The function definition is.

----------------------------------------------------------------------5.4

The alternating maxima and minima are identified by contour plot of Rastrigin's function.

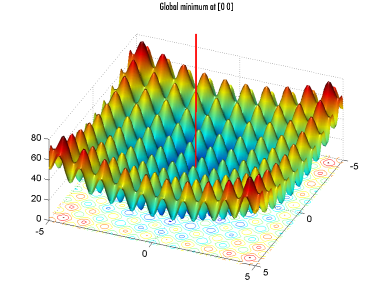
Test area is bounded to

Figure 10: Graphical overview Rastrigin’s function [[33](#HPo)]

* + 1. **Ackley’s function**

It is a multimodal function and is separable which is widely used. Originally this problem was defined for two dimensions, but the problem has been generalized to N dimensions. It has large number of local minima but the global minima are single. In MATLAB various implementations of this function are available. It is typical problem to solve with evolutionary algorithms. Its definition is given below.

-----------------------------5.5

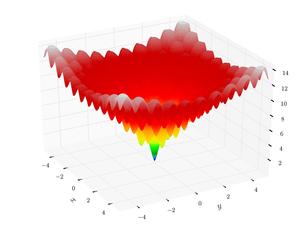


Figure 11: Graphical Overview of Ackley’s function [[33](#HPo)]

* + 1. **Schwefel’s Function**

Schwefel's function is deceptive in that the global minimum is geometrically distant over the parameter space from the next best local minima. Therefore the search algorithms are potentially prone to convergence in the wrong direction. The function definition is

-------------------------------------------------------------------5.6

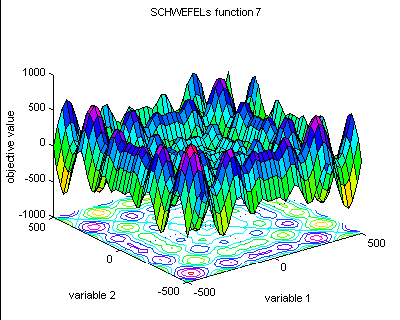


Figure 12: Graphical Overview of Schwefel’s Function [[33](#HPo)]

* + 1. **De Jong's function**

This is simplest test function and known as De Jong's function. This function is continuous, convex and uni-modal. This function is also called as sphere model.

----------------------------------------------------------------------------------------5.7

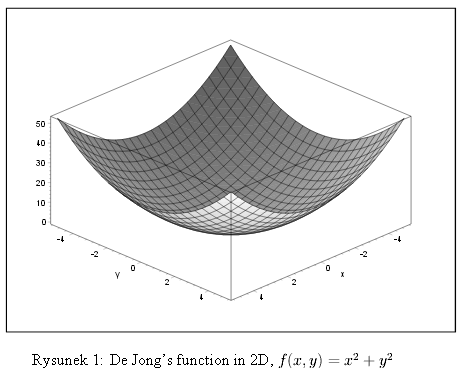


Figure 13: Graphical overview De Jong's function [[33](#HPo)]

* + 1. **Axis parallel hyper-ellipsoid function**

The axis parallel hyper-ellipsoid is same as De Jong function. The second name of this function is weighted sphere model. This Function is continuous, convex and unimodal. The definition of the function is as under

----------------------------------------------------------------------------------------5.8

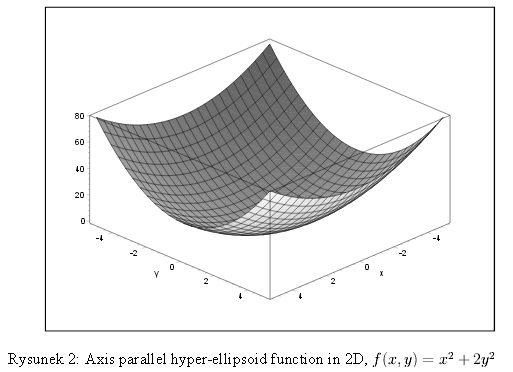


Figure 14: Graphical overview Axis parallel hyper-ellipsoid function [[33](#HPo)]

* 1. **Results**

Table 3. Comparative analysis of inertia weight strategies with dimensions=10 and particles=30

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S.No | Inertia Weight | Sphere | Griewank | Rosenbrock | Rastrigin | Ackley | Schwefel’s Function | De Jong’s Function | Axis Parallel Hyper-ellipsoid Function |
| 1 | Constant | 0.0428 | 0.3955 | 0.0275 | 318.6604 | 14.9933 | **0.00E+00** | 0.2933 | 2.625 |
| 2 | Random | 2.3559 | 0.5142 | 0.0029 | 338.9684 | 14.9918 | 3.03E-06 | 0.512 | 9.0654 |
| 3 | Adaptive Inertia weight | 0.046 | 0.6352 | 0.0369 | 327.1185 | 15.1202 | 1.81E-05 | 0.7594 | 23.2576 |
| 4 | Sigmoid Increasing Inertia weight | 5.90E-01 | 0.7741 | 0.6537 | 358.6249 | 15.0007 | 1.11E-04 | 0.2217 | 0.5446 |
| 5 | Sigmoid Decreasing Inertia weight | 0.0135 | 0.3444 | 0.2361 | 338.2048 | 14.9931 | 1.81E-05 | 0.187 | 0.3137 |
| 6 | Linear Decreasing Inertia weight | 0.0164 | 0.7545 | 0.0635 | 305.4208 | 14.9917 | 3.56E-05 | 0.0817 | 14.7297 |
| 7 | The Chaotic Inertia weight | 8.60E-02 | 0.725 | 0.1851 | 307.6016 | 14.9925 | 2.65E-04 | 0.769 | 1.6564 |
| 8 | Choitic Random Inertia weight | 0.1625 | 1.0437 | 0.0143 | 314.2576 | 14.9923 | 5.62E-05 | 0.9351 | 13.1375 |
| 9 | Oscillating Inertia weight | 0.0207 | 0.3314 | 0.4829 | 323.6689 | 14.9923 | 9.58E-05 | 0.3322 | 5.4248 |
| 10 | Global LocalBest Inertia weight | 2.5158 | 4.5352 | 7.5632 | 319.651 | 14.9943 | **0.00E+00** | 19.4401 | 123.9779 |
| 11 | Simulated Annealing Inertia weight | 0.1463 | 0.5969 | 0.1202 | 374.1946 | 15.0154 | 7.50E-04 | 0.4576 | 3.8448 |
| 12 | Natural Exponent Inertia weight (e1-pso) | 0.2696 | 0.3493 | 0.5147 | 337.1112 | 14.992 | 2.96E-06 | 0.8021 | 13.1831 |
| 13 | Natural Exponent Inertia weight (e2-pso) | 9.56E-02 | 0.0619 | 0.2233 | 304.4955 | 14.9955 | 1.72E-04 | 0.2412 | 3.7508 |
| 14 | Logrithem Decreasing Inertia weight | 0.081 | 0.7831 | **0.0007728** | 300.5982 | 14.9914 | 0.0029 | 0.0013 | 0.0043 |
| 15 | Exponent Decreasing Inertia weight | 0.047 | 0.5982 | 0.5474 | 336.0041 | 14.992 | 9.23E-07 | 0.1301 | 0.0179 |
| 16 | Proposed modified decreasing inertia weight | **0.0075** | **0.0343** | 0.0205 | **300.52** | **14.9907** | **0.00E+00** | **4.22E-04** | **0.000122** |
|  | BEST | 0.0075 | 0.0343 | 0.0007728 | 300.52 | 14.9907 | 0 | 0.000422 | 0.000122 |

Table 4: Comparative analysis of inertia weight strategies with dimensions=20 and particles=30

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S.No | Inertia Weight | Sphere | Griewank | Rosenbrock | Rastrigin | Ackley | Schwefel’s Function | De Jong’s Function | Axis Parallel Hyper-ellipsoid Function |
| 1 | Constant | 0.1299 | 0.1166 | 100 | 307.6746 | 15.2532 | 0.00E+00 | 0.853 | 0.0323 |
| 2 | Random | 0.0508 | 0.1883 | 100 | 339.1744 | 15.2524 | 3.03E-06 | 0.8354 | 0.0223 |
| 3 | Adaptive Inertia weight | 0.0027 | 0.4903 | 74.6195 | 326.3667 | 15.2522 | 1.81E-05 | 0.6349 | 1.0474 |
| 4 | Sigmoid Increasing Inertia weight | 0.0765 | 0.2003 | 100 | 304.3622 | 15.2521 | 1.11E-04 | 0.4132 | 0.039 |
| 5 | Sigmoid Decreasing Inertia weight | 0.0729 | 0.3446 | 100 | 338.4775 | 15.2518 | 1.81E-05 | 0.7099 | 0.0575 |
| 6 | Linear Decreasing Inertia weight | 0.001 | 0.1109 | 100 | 366.2207 | 15.2523 | 3.56E-05 | 0.5491 | 0.1675 |
| 7 | The Chaotic Inertia weight | 0.0144 | 0.1381 | 47.641 | 321.7651 | 15.2536 | 2.65E-04 | 0.7012 | 0.0343 |
| 8 | Choitic Random Inertia weight | 0.0374 | 0.2068 | 100 | 357.3803 | 15.2527 | 5.62E-05 | 0.1423 | 0.8106 |
| 9 | Oscillating Inertia weight | 0.1493 | 0.2563 | 100 | 324.861 | 15.2531 | 9.58E-05 | 0.4901 | 0.2055 |
| 10 | Global LocalBest Inertia weight | 3.8934 | 20.7765 | 100 | 591.2227 | 15.9516 | 0.00E+00 | 3.5204 | 12.7387 |
| 11 | Simulated Annealing Inertia weight | 0.1543 | 0.2481 | 100 | 317.9658 | 15.253 | 7.50E-04 | 0.6295 | 0.569 |
| 12 | Natural Exponent Inertia weight (e1-pso) | 0.0823 | 0.2177 | 100 | 402.3413 | 15.2522 | 2.96E-06 | 0.1432 | 0.5736 |
| 13 | Natural Exponent Inertia weight (e2-pso) | 0.0223 | 0.1164 | 100 | 344.9817 | 15.2525 | 1.72E-04 | 0.335 | 0.0809 |
| 14 | Logrithem Decreasing Inertia weight | 2.82E-04 | 0.0417 | **19.1611** | 320.0447 | 15.2516 | 0.0029 | 1.9634 | 0.0145 |
| 15 | Exponent Decreasing Inertia weight | 0.005 | 0.4021 | 100 | 304.1813 | 15.2529 | 9.23E-07 | 0.5114 | 0.7584 |
| 16 | Proposed modified decreasing inertia weight | **2.20E-04** | **0.0217** | 24.2527 | **300.4430** | **15.2102** | **0.00E+00** | **0.1415** | **0.00212** |
|  | BEST | 0.00022012 | 0.0217 | 19.1611 | 300.443 | 15.2102 | 0 | 0.1415 | 0.00212 |

Table 5: Comparative analysis of inertia weight strategies with dimensions=30 and particles=30

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S.No | Inertia Weight | Sphere | Griewank | Rosenbrock | Rastrigin | Ackley | Schwefel’s Function | De Jong’s Function | Axis Parallel Hyper-ellipsoid Function |
| 1 | Constant | 0.501 | 0.2706 | 658.1058 | 311.2786 | 15.322 | 6.45E-06 | 3.729 | 49.0727 |
| 2 | Random | 0.52 | 0.2078 | 312.522 | 344.946 | 15.3225 | 5.0796e-006 | 2.49 | 57.3436 |
| 3 | Adaptive Inertia weight | 0.2827 | 0.3067 | 213.6369 | 320.7591 | 15.3242 | 8.71E-05 | 1.8642 | 69.2388 |
| 4 | Sigmoid Increasing Inertia weight | 2.7348 | 0.2994 | 2057.7 | 318.9071 | 15.3219 | 1.14E-07 | 2.1934 | 9.2188 |
| 5 | Sigmoid Decreasing Inertia weight | 0.4199 | 0.8823 | 71.7482 | 321.7091 | 15.3224 | 2.50E-06 | 2.8493 | 1.5335 |
| 6 | Linear Decreasing Inertia weight | 0.509 | 0.5721 | **30.6434** | 310.3652 | 15.3227 | 3.41E-07 | 2.6742 | 18.1346 |
| 7 | The Chaotic Inertia weight | 0.4325 | 1.9433 | 618.6488 | 338.4968 | 15.3222 | 3.25E-05 | 3.427 | 7.2718 |
| 8 | Choitic Random Inertia weight | 0.847 | 0.3431 | 846.0524 | 403.9942 | 15.3228 | 2.43E-06 | 2.8671 | 80.7889 |
| 9 | Oscillating Inertia weight | 0.617 | 1.5475 | 75.061 | 410.6027 | 15.322 | 3.60E-05 | 3.9774 | 22.32 |
| 10 | Global LocalBest Inertia weight | 3 | 28.1051 | 2.28E+04 | 639.2709 | 16.0351 | 0.00E+00 | 46.352 | 61.4381 |
| 11 | Simulated Annealing Inertia weight | 0.67 | 0.962 | 1.88E+03 | 424.6981 | 15.3223 | 1.80E-04 | 8.2525 | 7.7213 |
| 12 | Natural Exponent Inertia weight (e1-pso) | 0.8122 | 1.2247 | 100.3868 | 430.7137 | 15.3227 | 4.03E-04 | 2.4944 | 113.8337 |
| 13 | Natural Exponent Inertia weight (e2-pso) | 0.735 | 0.5073 | 52.1101 | 335.7202 | 15.3219 | 7.47E-06 | 3.2462 | 113.1187 |
| 14 | Logrithem Decreasing Inertia weight | 2.70E-01 | 0.146 | 1.33E+03 | 346.4614 | 15.3232 | 0.00E+00 | 2.4021 | 144.3634 |
| 15 | Exponent Decreasing Inertia weight | 0.6728 | 0.4347 | 327.6496 | 304.7664 | 15.3228 | 7.24E-08 | 4.2568 | 84.2205 |
| 16 | Proposed modified decreasing inertia weight | **0.2623** | **0.1028** | 78.3735 | **300.4321** | **15.3219** | **0.00E+00** | **1.2765** | **0.1523** |
|  | BEST | 0.2623 | 0.1028 | 30.6434 | 300.4321 | 15.3219 | 0 | 1.2765 | 0.1523 |

* 1. **Results Analysis**

Detail Results of all the Benchmark functions using different inertia weights are given in tables 3, 4, and 5. Results are observed using dimensions 10, 20 and 30. Population size is 30 in all cases.

From the results shown in tables 3, 4, and 5, it can be observed that proposed inertia weight technique has high success rate in finding the optimal value as compared to other inertia weight techniques used in our experiment.

The proposed inertia weight technique looking dominant as compared to other inertia weight techniques and is more suitable to use as compared to others.

* 1. **Convergence Graphs**

Sphere function performs well with proposed inertia weight when dimensions are 10, 20 and30.

Figure 15: Sphere function convergence graph

Figure 16: Griewank Function convergence graph

Figure 17: Rosenbrock Function convergence graph

Figure 18: De Jong’s Function convergence graph

1. **Chapter 6 - Conclusion and Future work**
   * 1. **Conclusion**

In this research a novel modified decreasing inertia weight is proposed. A set of benchmark functions is used to perform the comparison of the existing techniques and the proposed technique. The research results are generated for various dimensions. Detailed results are given in chapter# 5 of this thesis. It can be observed that the Proposed modified Decreasing Inertia weight performs better and suitable than the other available inertia weight. The comparative analysis of proposed inertia weight technique with others shows that proposed inertia weight technique has high success rate for finding optimal value than other inertia weight techniques.

* + 1. **Future Work**

Future work of this paper may be to implement the proposed technique on remaining available benchmark function to generalize the conclusion about comparative analysis.

# Works Cited

|  |  |
| --- | --- |
| [1] | Wenjun Wang, Hui Wang, "An improved diversity-guided particle swarm optimization for numerical optimization ," . |
| [2] | J. Kennedy and R. Eberhart, "Particle Swarm Optimization," in *In Proceedings of IEEE International Conference on Neural Networks*, 1995, pp. 1942-1948. |
| [3] | Nguyen Quang Uy, Nguyen Xuan Hoai, R McKay, Pham Minh Tuan Pham Minh Tuan, "Initialising PSO with randomised low-discrepancy sequences," in *IEEE Congress on Evolutionary Computation*, 2007, pp. 1985-1992. |
| [4] | M. Pant, R. Thangaraj, and A. Abraham, "Particle Swarm Optimization Using Adaptive Mutation," in *19th International Conference on Database and Expert Systems Application*, 2008, pp. 519-523. |
| [5] | Z. J. a. A. R. B. H. Jabeen, "Opposition Based Initialization in Particle Swarm Optimization," in *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, 2009, pp. 2047-2052. |
| [6] | Chang Zhang, Zhiwei Ni, Zhangjun Wu, Lichuan Gu, "A Novel Swarm Model With Quasi-oppositional Particle," in *International Forum on Information Technology and Applications*, 2009. |
| [7] | Y Shi, R Eberhart, "A modified particle swarm optimizer," in *IEEE International Conference on Evolutionary Computation Proceedings IEEE World Congress on Computational Intelligence Cat No98TH8360*, 1998, pp. 69-73. |
| [8] | Qing Zhang,Changhe Li,Yong Liu,Lishan Kang, "Fast multi-swarm optimization with cauchy mutation and crossover operation," in *Proceedings of the 2nd international conference on Advances in computation and intelligence*, 2007, pp. 344-352. |
| [9] | S. Kai, S. Fan and J. M. Chang, "A Modified Particle Swarm Optimizer Using an Adaptive Dynamic Weight Scheme," in *Proceedings of the 1st international conference on Digital human modeling*, 2007, pp. 56-65. |
| [10] | C S Feng, S Cong, X Y Feng, "A new adaptive inertia weight strategy in particle swarm optimization," in *IEEE Congress on Evolutionary Computation*, 2007, pp. 786-790. |
| [11] | RF Malik, TA Rahman, SZM Hashim, R Ngah, "New Particle Swarm Optimizer with Sigmoid Increasing Inertia Weight," *International Journal of Computer Science and Security*, vol. 1, no. 2, 2007. |
| [12] | R C Eberhart, Yuhui Shi, "Tracking and optimizing dynamic systems with particle swarms," in *Proceedings of the 2001 Congress on Evolutionary Computation IEEE Cat No01TH8546*, 2001, pp. 94-100. |
| [13] | Yong Feng, Gui-Fa Teng, Ai-Xin Wang,Yong-Mei Yao, "Chaotic Inertia Weight in Particle Swarm Optimization," in *6th World Congress on Intelligent Control and Automation*, 2006, pp. 71-79. |
| [14] | Kyriakos Kentzoglanakis,Matthew Poole, "Particle Swarm Optimization with an Oscillating Inertia Weight," in *11th Annual conference on Genetic and evolutionary computation*, 2009. |
| [15] | W. Al-Hassan, MB Fayek, and SI Shaheen, "An optimized particle swarm technique for solving the urban planning problem," in *Computer Engineering and Systems*, 2006, p. 401–405. |
| [16] | Guimin Chen,Xinbo Huang,Jianyuan Jia, Zhengfeng Min, "Natural Exponential Inertia Weight Strategy in Particle Swarm Optimization," in *6th World Congress on Intelligent Control and Automation*, 2006, pp. 3672-3675. |
| [17] | Yue-lin Gao, Xiao-hui An,Jun-min Liu, "A Particle Swarm Optimization Algorithm with Logarithm Decreasing Inertia Weight and Chaos Mutation," in *International Conference on Computational Intelligence and Security*, 2008, pp. 61-65. |
| [18] | Hui-Rong LI,Yue-Lin GAO, "Particle swarm optimization algorithm with exponent decreasing inertia," in *Second International Conference on Information and Computing Science*, 2009. |
| [19] | E Ozcan, M Yllmaz, "Particle swarms for multimodal optimization," in *Adaptive and Natural Computing Algorithms*, 2007, pp. 366-375. |
| [20] | K. E. Parsopoulos , M. N. Vrahatis, "Initializing the particle swarm optimizer using the nonlinear simplex method," *Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*, 2002. |
| [21] | M. G. H. Omran and S.al-Sharhan, "Using Opposition-based Learning to improve the Performance of Particle Swarm Optimization," in *IEEE Swarm Intelligence Symposium*, 2008, pp. 1-6. |
| [22] | A. P. E. Frans van den Bergh, "A Cooperative Approach to Particle Swarm Optimization," *TRANSACTIONS AND EVOLUTIONARY COMPUTATION*, vol. 8, Jun. 2004. |
| [23] | Anthony Carlisle, Gerry Dozier, "Adapting Particle Swarm Optimization to Dynamic Environments," in *Design*, 2009, p. 429–434. |
| [24] | J.J. Liang, A.K Qin, "Comprehensive Learning Particle Swarm Optimizer for Global Optimization for Multimodal Functions," *TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, vol. 10, Jun. 2006. |
| [25] | S Kiranyaz, J Pulkkinen, M Gabbouj, "Multi-dimensional particle swarm optimization for dynamic environments," in *International Conference on Innovations in Information Technology*, 2008, pp. 34-38. |
| [26] | R. Brits .A.P. Engelbrecht, F. v. Den Bergh, "A Niching Particle Swarm Optimizer," in *In Proceedings of the Conference on Simulated Evolution and Learning*, 2002. |
| [27] | Morten Løvbjerg, Thomas Kiel Rasmussen, Thiemo Krink, "Hybrid particle swarm optimiser with breeding and subpopulations," in *Proc Genetic Evol Comp Conf*, 2001. |
| [28] | M Richards, D Ventura, "Dynamic sociometry in particle swarm optimization," in *Proceedings of the Sixth International Conference on Computational Intelligence and Natural Computing*, 2003. |
| [29] | Arlindo Silva,Ana Paula Neves F. da Silva,Ernesto Costa, "Chasing the Swarm: A Predator-Prey Approach to Function Optimisation," in *Evolutionary Optimization, PSO*, 2002. |
| [30] | Wei Wang,Zhiliang Wang, Xuejing Gu,Siyi Zheng , "Emotional Particle Swarm Optimization," *Emerging Intelligent Computing Technology and Applications. With Aspects of Artificial Intelligence*, vol. 5755, pp. 766-775, 2009. |
| [31] | F. Shahzad et al, "Opposition Based Particle Swarm Optimization with velocity clamping," in *advances in computational intelligence*, 339-348, p. 2009. |
| [32] | Srinivas Pasupuleti , Roberto Battiti, "The gregarious particle swarm optimizer (G-PSO)," in *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, 2006, pp. 67-74. |
| [33] | H. Pohlheim GEATbx. (www.geatbx.com), December 2006. |
| [34] | Jianbin Xin, Guimin Chen, Yubao Hai, "A Particle Swarm Optimizer with Multi-stage Linearly-Decreasing Inertia Weight," in *International Joint Conference on Computational Sciences and Optimization*, 2009, pp. 203-228. |
| [35] | L. Zhen-su, H. Zhi-rong and D.Juan, "Particle Swarm Optimization Using Sobol Mutation," in *Frontiers of electrical and electronic engineering,VOL. 1*, 2006, pp. 99-104. |
| [36] | H-R LI and Y-L Gao, "Particle swarm optimization algorithm with exponent decreasing inertia weight and stochastic mutation," in *Second International Conference on Information and Computing Science*, 2009, pp. 66-69. |
| [37] | Hui Wang,Hui Li, Yong Liu, Changhe Li, Sanyou Zeng, "Opposition-based particle swarm algorithm with cauchy mutation," in *IEEE Congress on Evolutionary Computation*, 2007, pp. 4750-4756. |
| [38] | Hemlata S.Urade, Prof Rahila Patel, "Study and Analysis f Particle Swarm Optimization: A Review," in *2nd International Conference on Information and Communication Technology*, 2011. |
| [39] | YAN Chun-man, GUO Bao-long, WU Xian-xiang, "Empirical Study of Inertia Weight Particle Swarm Optimization with Constraint Factor," *International Journal of Soft Computing and Software Engineering*, vol. 2, 2012. |
| [40] | Hui-Rong LI, Yue-Lin GAO, "Particle swarm optimization algorithm with expnent decreasing inertia weight and stochastic mutation," in *2nd International Conference on Information and Computing Science*, 2009. |
| [41] | M Imran, H Jabeen, M Ahmad, Q Abbas, W Bangyal, "Opposition based PSO and mutation operators," in *Education Technology and Computer ICETC 2010 2nd International Conference*, 2010, pp. 506-508. |
| [42] | X. Wu and M. Zhong, "Particle Swarm Optimization Based on Power Mutation," in *ISECS International Colloquium on Computing, Communication, Control, and Management*, 2009, pp. 464-467. |
| [43] | A Chatterjee, P Siarry, "Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization," in *Computers & Operations Research*, 2006, pp. 859-871. |
| [44] | Jian Hu, Zhiqiang Wang, Shaojie Qiao, JianChao Gan, "The fitness evaluation strategy in particle swarm optimization," *Applied Mathematics and Computation*, vol. 217, no. 21, p. 8655–8670, Jul. 2011. |
| [45] | Ahmad Nickabadi,Mohammad Mehdi Ebadzadeh, "A novel particle swarm optimization algorithm with adaptive inertia weight," *Applied Soft Computing*, vol. 11, no. 4, p. 3658–3670, Jun. 2011. |
| [46] | Farrukh Shahzad,A. Rauf Baig,Sohail Masood,Muhammad Kamran,Nawazish Naveed , "Opposition-Based Particle Swarm Optimization with Velocity Clamping (OVCPSO)," *Advances in Intelligent and Soft Computing*, vol. 116, pp. 339-348, 2009. |
| [47] | Jun Tang, Xiajuan Zhao, "Particle Swarm Optimization with Adaptive Mutation," in *Information Engineering, WASE International Conference*, 2009, pp. 234-237. |
| [48] | James Blondin, "Particle Swarm Optimization: A Tutorial," in *Armstrong Atlantic State Book*, 2009. |
| [49] | P. Andras, *A Bayesian Interpretation of the Particle Swarm Optimization and Its Kernel Extension*. November , 2012. |
| [50] | Hui Wanga, "Diversity enhanced particle swarm optimization with neighborhood," in , February 2013, p. 119–135. |
| [51] | Pehlivanoglu, Y.V, "A New Particle Swarm Optimization Method Enhanced With a Periodic Mutation Strategy and Neural Networks," in *IEEE Transactions on Evolutionary Computation*, June 2013 . |
| [52] | Swapnil Gaul , "Particle Swarm Optimization and it's Applications in Electromagnetics," in , Jan, 2012. |
| [53] | Stephen Chen, "Particle Swarm Optimization with pbest cross over," in , June, 2012. |