A Non-deterministic Adaptive Inertia Weight in PSO

Kusum Deep
Department of Mathematics
Indian Institute of Technology
Roorkee
Uttarakhand, India.
kusumfma@iitr.ernet.in

Madhuri
Department of Mathematics
Indian Institute of Technology
Roorkee
Uttarakhand, India.
madhuriiitr@gmail.com

Jagdish Chand Bansal
Department of Mathematics
ABV-Indian Institute of Information
Technology and Management
Gwalior (M.P.), India.
jcbansal@iiitm.ac.in

ABSTRACT
Particle Swarm Optimization (PSO) is a relatively recent swarm intelligence algorithm inspired from social learning of animals. Successful implementation of PSO depends on many parameters. Inertia weight is one of them. The selection of an appropriate strategy for varying inertia weight *w* is one of the most effective ways of improving the performance of PSO. Most of the works done till date for investigating inertia weight have considered small values of *w*, generally in the range [0,1]. This paper presents some experiments with widely varying values of *w* which adapts itself according to improvement in fitness at each iteration. The same strategy has been implemented in two different ways giving rise to two inertia weight variants of PSO namely Globally Adaptive Inertia Weight (GAIW) PSO, and Locally Adaptive Inertia Weight (LAIW) PSO. The performance of the proposed variants has been compared with three existing inertia weight variants of PSO employing a test suite of 6 benchmark global optimization problems. The experiments show that the results obtained by the proposed variants are comparable with those obtained by the existing ones but with better convergence speed and less computational effort.

Categories and Subject Descriptors
I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search–*heuristic methods.*

General Terms
Experimentation.

Keywords
Particle Swarm optimization, Dynamic inertia weight, Adaptive inertia weight, Non-deterministic inertia weight.

1. INTRODUCTION
PSO (Particle Swarm Optimization) is a population based stochastic optimization technique. It is a newer addition to the class of nature inspired optimization algorithms. It was introduced by Kennedy and Eberhart in 1995 [9, 10]. Only within a few years of its introduction PSO has gained wide popularity as a powerful global optimization tool and is competing with well-established population based evolutionary algorithms like GA [5]. In the recent years it has attracted a lot of attention from researchers all over the world and has been successfully applied for solving a wide variety of optimization problems occurring in diverse fields of science, engineering and industry [14, 15, 18].

GECCO'11, July 12–16, 2011, Dublin, Ireland.

The fundamental idea behind PSO is the mechanism by which the birds in a flock and the fishes in a school cooperate while searching for food. In PSO terminology population is called swarm and the individual solutions are referred to as particles. Each particle in the swarm relies on its own experience as well as the experience of its best neighbor. Each particle has an associated fitness value. These particles move through search space with a specified velocity in search of optimal solution. Each particle maintains a memory which helps it in keeping the track of the best position it has achieved so far. This is called the particle's personal best position (pbest) and the best position the swarm has achieved so far is called global best position (gbest). PSO has two primary operators: Velocity update and Position update. During each generation each particle is accelerated towards the gbest and its own pbest. At each iteration a new velocity value for each particle is calculated according to the following velocity update equation:

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

The new velocity value is then used to calculate the next position of the particle in the search space, according to the following position update equation:

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

This process is then iterated until a predefined stopping criterion is satisfied. Here $X_i=(x_{i1},x_{i2},...x_{id})$ represents the position of the i^{th} particle in a d-dimensional search space, $P_{besti}=(p_{i1},p_{i2},...p_{id})$ is i^{th} particle's pbest position, $P_{gbest}=(p_{g1},p_{g2}....p_{gd})$ is gbest position and $V_i=(v_{i1},v_{i2},...v_{id})$ is the velocity of i^{th} particle. The acceleration coefficients c_1 and c_2 control how far a particle can move in a single iteration. Typically, these are both set equal to a value of 2. The coefficients r_1 and r_2 are the uniformly generated random numbers in the range [0, 1] to provide stochastic nature to the algorithm.

In PSO the search for global optimal solution is accomplished by maintaining a dynamic balance between exploration and exploitation. Exploration is the ability of an algorithm to explore different regions of the search space in order to locate a good optimum. Exploitation, on the other hand, is the ability to concentrate the search around a promising area in order to refine a candidate solution [4]. So an optimal balance between exploration and exploitation is the key to the performance control of PSO. To control the global exploration of particles the concept of inertia weight was introduced by Shi and Eberhart [16]. The inertia

weight controls the momentum of a particle by weighing the contribution of the previous velocity on the new velocity according to the following equation:

$$v_{id} = \underbrace{wv_{id}}_{\text{Inertia component}} + \underbrace{c_1 r_1 (p_{id} - x_{id})}_{\text{Cognitive component}} + \underbrace{c_2 r_2 (p_{gd} - x_{id})}_{\text{Social component}} \quad (3)$$

Where w is the inertia weight. Clearly, with large values of inertia weight particles will have large position updates and so the swarm will have more diversity and hence more exploration capability which means more exploration of new search areas in pursuit of a better solution. On the other hand smaller values of inertia weight will lead to less variation in velocity which provides slower updating for fine tuning a local search. However, with zero inertia (w=0) there will be quick changes in the directions of particles and they will search locally around their current position. It has been inferred that the system should start with a high inertia weight for global exploration and then it should decrease successively to facilitate finer local exploitations. This helps the system to approach the optimum of the fitness function quickly.

Since the introduction of inertia weight in PSO, a lot of research has been devoted to find the optimal or the standard value of the inertia weight. But results show that its value is problem dependent and no standard value has yet been found. It has been established that a reasonable choice would be to vary the inertia weight over iterations instead of using a fixed value of it during the course of a run. Many researchers have proposed a lot of strategies for dynamically adjusting inertia weight. Some of them are random inertia weight [17], fuzzy based adaptive inertia weight [3], increasing inertia weight [19, 20], decreasing inertia weight [2]. Hu and Zeng [6] proposed three ways of dynamically adjusting inertia weight, two of these were linear and one was non-linear. A number of other ways for non-linearly adapting inertia weight have been proposed in [1, 7, 8, 13].

Almost all the methods (except a few) that have been proposed till date, for the dynamic adjustment of inertia weight, have used some deterministic approach and have taken its value between 0 and 1. In this paper a non-deterministic way of dynamically adjusting the inertia weight is proposed and larger values of inertia weight have also been allowed. This method is based on the improvement in the fitness of the particles as the search process progresses. The performance of the proposed PSO model is compared with some other available PSO models reported in the literature.

The remainder of this paper is organized as follows: The second section of the paper describes the proposed strategy for dynamic adjustment of inertia weight. Computational results obtained for the test functions are presented in third section. Finally we conclude the work in section 4.

2. PROPOSED INERTIA WEIGHT PSO

This paper proposes a new approach for adjusting inertia weight in PSO that adapts itself at each iteration according to the improvement in best fitness. Here the inertia weight has been taken as a function of iteration number and is updated according to the following equation:

$$\left.\begin{array}{ll} w(t+1) = 0.9, & \text{if } t = 0; \\ w(t+1) = f(t) - f(t-1) & \text{if } t > 0 \end{array}\right\} \quad (4)$$

Where $w(t+1)$ is the inertia weight at $(t+1)^{th}$ iteration and $f(t)$ is the best fitness value at t^{th} iteration. This modification increases the influence of potentially fruitful inertial directions while decreasing the influence of potentially unfavourable inertial directions. Clearly, this way of adapting the inertia weight may sometimes lead to very large values of w resulting in the explosion of particle's velocities, therefore the velocities are clamped in the range $[-V_{dmax}, V_{dmax}]$ to keep the particles within the boundaries of the search space.When using this approach, the variations of inertia weight as obtained experimentally are as shown in the Figure 1 for two benchmark problems.

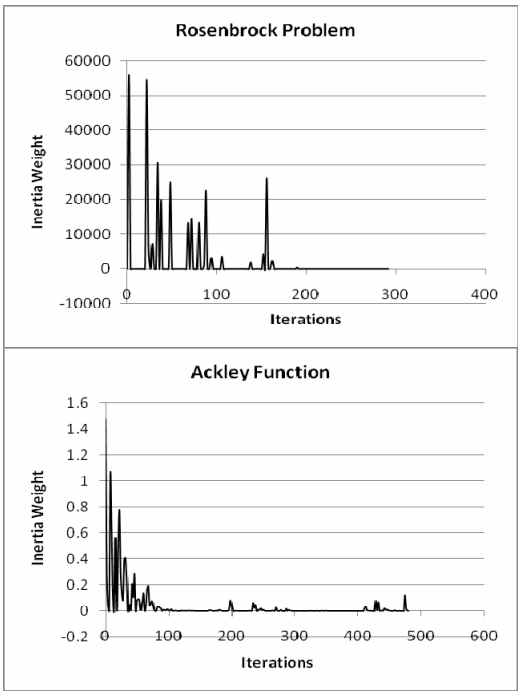

Figure 1. Variation of inertia weight with iterations for Rosenbrock, and Ackley Functions

Here the swarm starts with w=0.9 which is then updated using equations (4) according as the improvement in global best fitness. For all the functions used in our study it is observed that there are large oscillations in the value of inertia weight in the initial iterations which help the swarm in maintaining the diversity resulting in good exploration. So the particles can fly through the total search space quickly. Towards the end it is observed that the oscillations become smaller and smaller which facilitate fine tuning of the final solution. Based on this observation, we can expect that this strategy may perform well for enhancing the performance of PSO. From Figure 1 another observation is that during the search the inertia weight sometimes becomes zero and does not increase for many consecutive iterations which means that the swarm sometimes stagnates at a suboptimal solution. The smaller the inertia weight, the more do the cognitive and social components control position updates. With zero inertia of swarm, stagnation for many consecutive iterations implies that the social and cognitive components are not capable of easily escaping the suboptimum. It slows down the search process. To overcome this situation it is proposed that if the swarm stagnates for M consecutive iterations, the swarm should be provided with some inertia to increase the diversity. So the inertia weight equation is modified as follows:

$$\left.\begin{array}{l} w(t+1) = 0.9 \qquad\qquad if\ t = 0 \\ and \quad for\ t > 0 \quad we\ have \\ w(t+1) = f(t) - f(t-1) \\ w = w_{start} - (w_{start} - w_{end}) * t / t_{max}; \qquad if\ w = 0\ for\ M \\ \qquad\qquad\qquad\qquad\qquad\qquad successive\ iterations \end{array}\right\} \quad (5)$$

Where w_{start} is the initial value of the inertia weight, and w_{end} is the final value of the inertia weight, and t is the current iteration (generation) of the algorithm while t_{max} is the maximum number of iterations (generations) specified by the user. This strategy may help in decreasing the period of entrapment (i.e., the number of iterations for which the swarm stagnates) in suboptimal solutions during the search and hence improve the convergence rate. We use this strategy in two ways, namely, globally and locally. In the global strategy each particle in the swarm has the same inertia weight that updates according to equations (5) using the improvement in global best fitness. It is called globally adaptive inertia weight (GAIW). At any iteration if the global best fitness improves, the particles are encouraged to search in their current directions, otherwise inertia is made zero and the swarm starts contracting to the current global best position until another particle takes over or until the swarm is provided with some inertia from which time it starts globally exploring the search space. If the global best particle stops moving for a few iterations then the whole swarm may stop changing. It may lead to premature convergence. Considering this possible disadvantage of the global strategy a local strategy is proposed i.e., the inertia weight for each particle at each iteration is updated individually according to equations (5) using the improvement in its personal best fitness. It is called locally adaptive inertia weight (LAIW). At any iteration if a particle's personal best fitness improves, the particle is encouraged to search in its current direction, otherwise its inertia is made zero and the particle starts searching locally until its personal best improves or until it is provided with some inertia from which time it starts global exploration. Also since the inertia weight of each particle is updated individually, all the particles in the swarm possibly have different inertia weight. Therefore some particles may search globally while the others are searching locally. This leads to automatic balancing of local and global search. This strategy strongly avoids entrapment in suboptimal solutions and due to increased diversity it is highly explorative.

3. COMPUTATIONAL EXPERIMENTS

The proposed inertia weight variants of PSO i.e., GAIW and LAIW are compared with three existing inertia weight variants namely fixed inertia weight (FIW), linearly decreasing inertia weight (LDIW) and non-linearly decreasing inertia weight (NDIW). The equations (6) and (7) are used to determine LDIW [16] and NDIW [11] respectively.

$$w = w_{start} - (w_{start} - w_{end}) * t / t_{max} \quad (6)$$

$$w = (w_{start} - w_{end}) * tan\left(\frac{7}{8} * \left(1 - \left(\frac{t}{t_{max}}\right)^{k}\right)\right) + w_{end} \quad (7)$$

Where w_{start}, w_{end}, t_{max} and t have the same meanings as in equations (5), *tan()* is the trigonometric tangent function, and k is the control variable which can control the smoothness of the curve that reflects the relationship between the w and t.

3.1 Test Problems

The relative performance of the algorithms is evaluated on a set of 6 benchmark problems. These problems are of continuous variables and have different degree of complexity and multimodality. The problem size for all problems is kept fixed at 30. All these problems are of minimization type. The problem set is shown in Table 1.

3.2 Parameter selection

General parameter setting is used for experiments. We use c_1=c_2=2, swarm size=60, t_{max}=5000. The maximum allowable velocity in each dimension has been taken to be $V_{dmax} = (x_{dmax} - x_{dmin}) * 0.25$, where x_{dmax} and x_{dmin} are the upper and lower bounds for particles' positions in d^{th} dimension of the search space. All these settings have been kept same for all the algorithms considered here. The termination criterion for all algorithms is a combination of the following two conditions: (i) reaching the maximum number of iterations, (ii) getting a solution within the tolerance limit ε (see Table 1), which means that simulation of an algorithm is stopped as soon as either of these conditions is satisfied. For FIW PSO w=0.68 is set. For all other algorithms w_{start}=0.9 and w_{end}=0.4 are set. Also for NDIW PSO k=0.6 (as recommended in [11]) is taken. For GAIW and LAIW M=25 is taken for all problems.

3.3 Performance Evaluation Criteria

In order to avoid attributing the results to the choice of a particular initial population, each test is performed 100 times, starting from various randomly selected points in the search space. All the PSOs are implemented in C and experiments are carried out on a Xeon, 3.4 GHz machine under LINUX operating system. All the results that have been recorded and presented here have been averaged over the successful runs out of 100. A run is considered a success if the algorithm finds a solution satisfying $|f_{opt} - f_{min}| < \varepsilon$, where f_{min} is the best solution found when an algorithm terminates and f_{opt} is the known global optimum of the problem. For each algorithm and for each problem the following are recorded:

1. Average number of function evaluations of successful runs (AFE).
2. Average Execution Time (AET) of successful runs.
3. Success Rate (SR) = $\frac{(\#\ of\ successful\ runs)}{total\ runs} \times 100$
4. Average Error (AE) = Average of $|f_{opt} - f_{min}|$ over successful runs.
5. Standard Deviation (SD) = Standard deviation of the error $|f_{opt} - f_{min}|$.
6. Success Performance (SP)
$$= \frac{(AFE)}{\#\ of\ successful\ runs} \times (\#\ of\ total\ runs) \quad [12]$$

These measures are shown in tabular as well as graphical form using box-plots.

3.4 Results and Discussions

All the results are recorded in Table 2. In Figures 2 the best performing PSO is marked with star. The box-plots have been drawn for all the functions taken together. The goal of the

analysis is to observe if the proposed strategy shows an improvement over the existing ones or not.

AFE is a measure of the computational cost of the method. It is clear from the results that LAIW performs the best from AFE point of view. So the order of PSOs based on the computational cost is:

[1]LAIW > GAIW > NDIW > FIW > LDIW.

Thus the proposed PSOs significantly reduce the computational effort.

AET is a measure of the convergence rate of the method. The results clearly show that from this point of view also LAIW performs the best. The order of PSOs based on AET is:

LAIW > GAIW > FIW > NDIW > LDIW

Success rate is a measure of the reliability of the method. It is clear that performance of LAIW is again the best among all PSOs considered. Further the order of PSOs based on the SR performance is:

LAIW > GAIW > NDIW > FIW > LDIW

except for Ackley function, where GAIW gives better SR than LAIW.

In order to observe the consolidated effect on SR and AFE performance, a comparison among all five versions is made on the basis of success performance (SP) also. From SP point of view following order is seen (except for Ackley function):

LAIW > GAIW > NDIW > FIW > LDIW.

Now the most accurate method is sought. For this the comparison based on average error (AE) and standard deviation (SD) of successful runs is carried out. Standard deviation gives the information about the consistency of the optimal solution over the successful runs. Smaller value of SD indicates the consistency of the algorithm in finding the optimal solution. From this point of view the following performance order of the algorithms is observed:

LAIW > GAIW > FIW > NDIW > LDIW.

On the basis of above analysis, LAIW gives overall best performance among all five versions of PSO considered here. So it may be concluded that the proposed strategy gives significantly better results than the existing ones.

Let us now discuss the possible reasons for the good performance of proposed strategy. In standard PSO each particle moves under the influence of three velocity components. When inertia weight is kept fixed or is varied between 0 and 1, each of the three velocity components affect, almost equally, the movement of the particle during entire search process. But in the proposed strategy when the value of w is very large, the particles perform almost individual search controlled mainly by inertia, and when the value of w is very small, the search is controlled mainly by the social and cognitive components. For intermediate values of w there is a balance between the two. In this way the search pattern becomes like a combination of individual search and social cooperation with their due weightages that vary with iterations. During individual search the swarm has high diversity which is an important factor for good performance of any population based optimization algorithm. By studying the patterns of variation of inertia weight for various functions used here, we see that during the initial iterations w takes very large values, so initially individual search is more effective than social search.

Table 1: Description of Test Functions

Sl.	Name	Function	Bounds	ε
1	Sphere	$\sum_{i=1}^{n} x_i^2$	$[-5.12,5.12]^{30}$	0.001
2	Griewank	$1+\frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)$	$[-600,600]^{30}$	0.001
3	Rosenbrock	$\sum_{i=1}^{n}(100(x_{i+1}-x_i^2)^2+(x_i-1)^2)$	$[-30,30]^{30}$	100
4	Rastrigin	$10n+\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i))$	$[-5.12,5.12]^{30}$	50
5	Ackley	$-20\exp\left(-0.02\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+e$	$[-30,30]^{30}$	0.001
6	Schwefel 3	$\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	$[-10,10]^{30}$	0.001

[1] A > B implies that algorithm A performs better than algorithm B for that particular point of view.

Table 2: Results of Experiments

Performance Measure	Function Name	FIW	LDIW	NDIW	GAIW	LAIW
Average functional Evaluations	Sphere	70921	161621	99573	15080	1560
	Griewank	157472	187242	124739	36272	1867
	Rosenbrock	121825	186560	120594	140243	1480
	Rastrigin	64856	180404	96820	45017	651
	Ackley	177488	194984	131728	72143	119713
	Schewfel 3	177488	180559	119416	42372	1980
Average Execution Time	Sphere	0.3032	0.6684	0.5024	0.0778	0.0088
	Griewank	1.0962	1.3029	0.9827	0.2905	0.0159
	Rosenbrock	0.6173	0.9569	0.7282	0.8489	0.0099
	Rastrigin	0.5989	1.0917	0.6728	0.3147	0.0048
	Ackley	1.1561	1.2461	0.9629	0.5368	0.9412
	Schewfel 3	0.6782	0.9604	0.7439	0.2702	0.0137
Success Rate	Sphere	100	100	100	100	100
	Griewank	21	31	39	17	100
	Rosenbrock	74	53	75	94	100
	Rastrigin	81	68	94	98	100
	Ackley	97	97	100	99	64
	Schewfel 3	81	55	91	15	100
Average Error	Sphere	0.000953	0.000945	0.000942	0.000934	0.000114
	Griewank	0.000954	0.000924	0.000925	0.000885	0.000007
	Rosenbrock	98.34058	97.89938	98.88599	94.54052	32.66543
	Rastrigin	49.21176	49.39174	49.19422	49.4536	4.322524
	Ackley	0.000971	0.000966	0.000965	0.000928	0
	Schewfel 3	0.000948	0.000954	0.000964	0.000799	0
Standard Deviation	Sphere	0.000034	0.000057	0.000068	0.000065	0.000126
	Griewank	0.001852	0.001381	0.001158	0.00196	0.000067
	Rosenbrock	58.35779	92.35525	57.10645	28.01283	11.49765
	Rastrigin	23.86082	33.89431	12.4941	7.110318	7.890891
	Ackley	0.000173	0.000173	0.000029	0.00015	0
	Schewfel 3	0.000462	0.000863	0.000305	0.001913	0
Success Performance	Sphere	70921	161621	99573	15080	1560
	Griewank	749866.7	604006.5	319843.6	213364.7	1867
	Rosenbrock	164628.4	352000	160792	149194.7	1480
	Rastrigin	80069.1	265300	103000	45935.71	651
	Ackley	182977.3	201014.4	131728	72871.72	187051.6
	Schewfel 3	219121	328289.1	131226.4	282480	1980

Consequently the swarm is more diverse and performs better exploration rapidly. So the good regions of search space are quickly identified during first few iterations. In the later iterations the values of inertia weight are relatively small so the social cooperation is given more weightage now, and usual PSO search is performed to exploit the information obtained yet.

The overall search pattern may be viewed as a swarm starting with zero velocities and w=0.9 first explore the search space vigorously, sometimes searching locally around good positions and in the later iterations concentrate the search in good areas found so far.Hence it can be said that the faster convergence of proposed PSO is due to (i) the quick exploration in first few iterations, and (ii) the reduction in the period of entrapment during the search. Also the high success rate may be due to the increased diversity of the swarm because now the swarm explores the search space more effectively.

4. CONCLUSIONS

In this paper two variants of PSO have been proposed, which are based on a new approach for dynamically adjusting the inertia weight at each iteration. The proposed variants are tested on 6 benchmark problems. The aim was to increase diversity of swarm for more exploration of the search space during initial iterations and apply mild fine tuning during later iterations so that the optimal solution could be approached with better accuracy, simultaneously making the PSO capable of avoiding entrapment in suboptimal solutions and, also improving its convergence rate. All these purposes are simultaneously and successfully fulfilled to a satisfactory level by using the proposed inertia weight strategy, as is clear from the results of experiments. An advantage of using the proposed strategy is that it is almost problem independent i.e., the same strategy gives different inertia weights for different problems according to that problem's requirement.

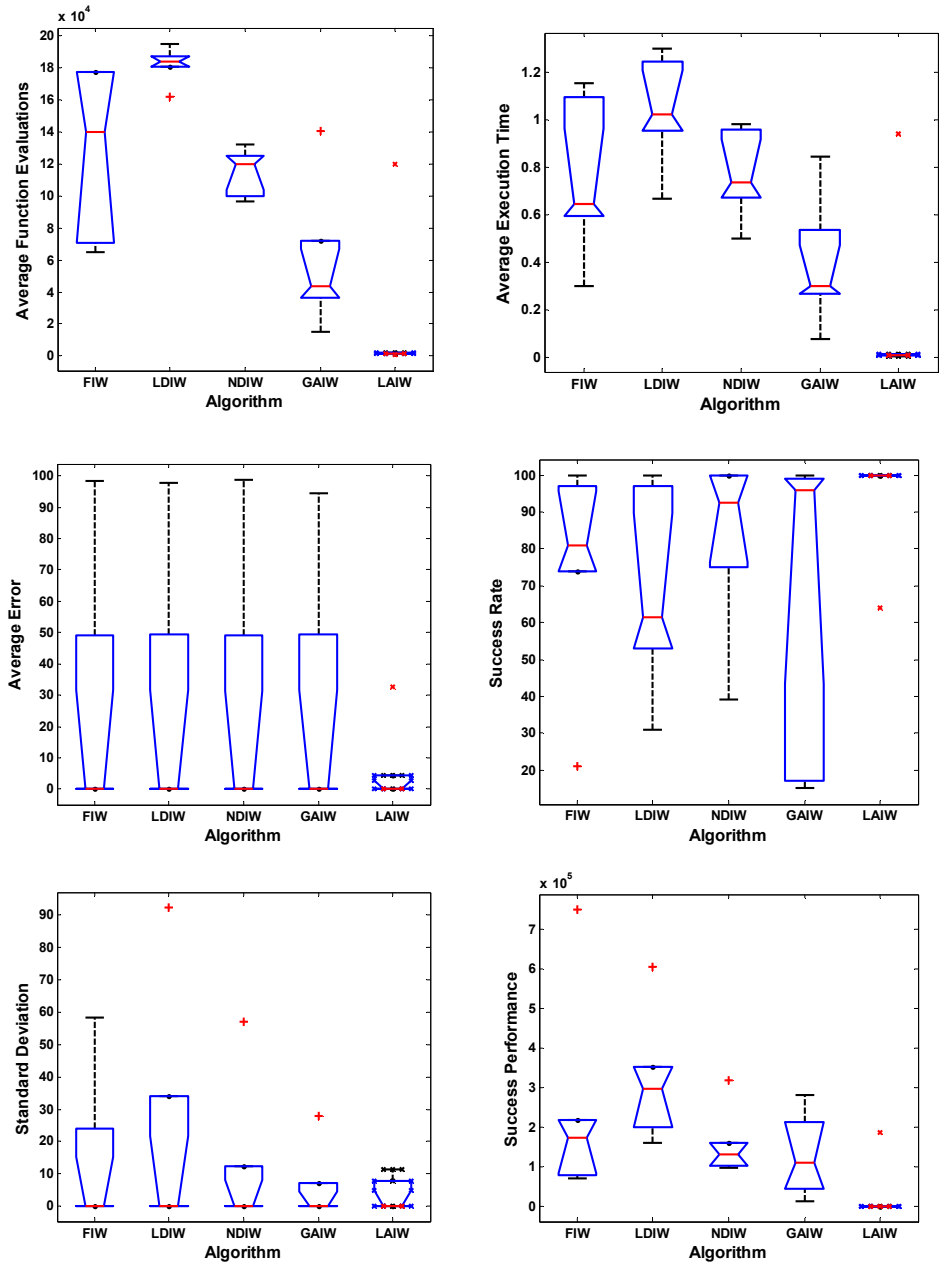

Figure 2: Box plots for various performance measures

Thus it can be concluded that the proposed inertia weight variants increase the performance of PSO significantly and can be used for different kinds of optimization problems thus releasing the user from the pain of indulging into extensive experiments for finding an appropriate setting of inertia weight.

5. ACKNOWLEDGMENTS

The second author , Madhuri, acknowledges Council of Scientific and Industrial Research, New Delhi, India, for providing the financial support for this work.

6. REFERENCES

[1] Chatterjee, A., and Siarry, P. Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization. *Computers & Operations Research, 33* 2006, 859–871.

[2] Cui, H. M., and Zhu, Q. B. Convergence analysis and parameter selection in particle swarm optimization. *Computer Engineering and Applications, 23*, 43, 2007, 89-91.

[3] Eberhart, R. C., and Shi, Y. 2001. Tracking and optimizing dynamic systems with particle swarms. In Proceeding Congress on Evolutionary Computation 2001, Seoul, Korea, Piscataway, NJ: IEEE Service Centre.

[4] Engelbercht, A. P. Fundamentals of computational swarm intelligence. John Wiley & Sons, 2005.

[5] Goldberg, D. E. Genetic Algorithms in Search Optimization, and Machine Learning, Reading MA: Addison-Welsey, 1989.

[6] Hu, J. X., and Zeng, J.C. Selection on Inertia Weight of Particle Swarm Optimization. *Computer Engineering*, *33*, 11, June 2007, 193-195.

[7] Hu, J. Z., Xu, J., Wang, J. Q., and Xu, T. Research on Particle Swarm Optimization with dynamic inertia weight. In *Proceedings of International Conference on Management and Service Science,* (*MASS '09*), 2009.

[8] Jiao, B., Lian, Z., and Gu, X. A dynamic inertia weight particle swarm optimization algorithm. *Chaos, Solitons and Fractals*, 37, 2008, 698–705.

[9] Kennedy, J., and Eberhart, R. C. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, WA Australia, 1995, 1942–1948.

[10] Kennedy, J., Eberhart, R. C., and Shi Y. *Swarm intelligence.* Morgan Kaufmann Publishers, 2001.

[11] Li, L., Bing, X., Ben, N., Lijing, T., and Jixian, W. *A Novel Particle Swarm Optimization with Non-linear Inertia Weight Based on Tangent Function*. D.-S. Huang et al. (Eds.): ICIC 2009, LNAI, 5755, Springer-Verlag Berlin Heidelberg, 2009, 785–793.

[12] Liang, J., Runarsson, T., Mezura-Montes, E., Clerc, M., Suganthan, P., Coello, C. and Deb, K. *Problem definitions and evaluation criteria for the CEC 2006*, Special Session on Constrained Real-Parameter Optimization, Technical Report, 2006.

[13] Malik, R. F., Rahman, T. A., Hashim, S. Z. T., and Ngah, R. New Particle Swarm Optimizer with Sigmoid Increasing Inertia weight, *International Journal of Computer Science and Security, 1,* 2, 2007.

[14] Mendes, R., Cortez, P., Rocha, M., and Neves, J. Particle Swarms for Feedforward Neural Network Training. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2002)*, 2002, 1895–1899.

[15] Parsopoulos, K. E., Papageorgiou, E. I., and Groumpos, P.P. A First Study of Fuzzy Cognitive Maps Learning Using Particle Swarm Optimization. In *Proceedings of IEEE Congress on Evolutionary Computation 2003 (CEC 2003)*, Canbella, Australia, 2003, 1440–1447.

[16] Shi, Y., and Eberhart, R. C. A modified particle swarm optimizer. In *Proceedings of the IEEE Conference on Evolutionary Computation,* Piscataway, NJ, IEEE Press, 1998, 69-73.

[17] Shi, Y., and Eberhart, R. C.. Fuzzy Adaptive Particle Swarm Optimization. In *Proceedings of Congress on Evolutionary Computation.* Seoul, Korea, Piscataway, NJ, IEEE Service Centre, 2001, 101-106.

[18] Venayagamoorthy, G. K., and Doctor, S. Navigation of Mobile Sensors Using PSO and Embedded PSO in a Fuzzy Logic Controller. In *Proceedings of the 39th IEEE IAS Annual Meeting on Industry Applications*, Seattle, USA, 2004, 1200–1206.

[19] Zheng, Y. L., Ma, L. H., Zhang L. Y., and Qian, J. X. Empirical Study of Particle Swarm Optimizer with an Increasing Inertia Weight. In *Proceeding of the IEEE Congress on Evolutionary Computation*, Vol.1, 2003, 221-226.

[20] Zheng, Y. L., Ma, L. H., Zhang L. Y., and Qian, J. X. On the convergence analysis and parameter selection in particle swarm optimization. In *Proceedings of the Second International Conference on Machine Learning and Cybernetics*, 2003, 1802-1807.