A nature inspired adaptive inertia weight in particle swarm optimisation

Madhuri Arya* and Kusum Deep

Department of Mathematics,
Indian Institute of Technology Roorkee,
Roorkee, Uttarakhand 247667, India
E-mail: madhuriiitr@gmail.com
E-mail: kusumfma@iitr.ernet.in
*Corresponding author

Jagdish Chand Bansal

Department of Applied Mathematics,
Faculty of Mathematics and Computer Science,
South Asian University,
New Delhi, India
E-mail: jcbansal@sau.ac.in

Abstract: The selection of an appropriate strategy for adjusting inertia weight *w* is one of the most effective ways of enhancing the performance of particle swarm optimisation (PSO). Recently, a new idea, inspired from social behaviour of humans, for adaptation of inertia weight in PSO, has been proposed, according to which *w* adapts itself as the improvement in best fitness at each iteration. The same idea 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. In this paper, the performance of these two variants has been compared with three other inertia weight variants of PSO employing an extensive test suite of 15 benchmark global optimisation problems. The experimental results establish the supremacy of the proposed variants over the existing ones in terms of convergence speed, and computational effort. Also, LAIW PSO comes out to be the best performer out of all the algorithms considered in this study.

Keywords: adaptive inertia weight; dynamic inertia weight; particle swarm optimisation; PSO; nature inspired inertia weight.

Reference to this paper should be made as follows: Arya, M., Deep, K. and Bansal, J.C. (2014) 'A nature inspired adaptive inertia weight in particle swarm optimisation', *Int. J. Artificial Intelligence and Soft Computing*, Vol. 4, Nos. 2/3, pp.228–248.

Biographical notes: Madhuri Arya is a PhD student. She received her Master's in Mathematics in 2006. She is doing her PhD on particle swarm optimisation and its applications. Her research interests are optimisation, nature inspired computing, and their applications in health, medicine and social sciences.

Kusum Deep is a Professor in Department of Mathematics, Indian Institute of Technology Roorkee, India. She holds a Post Doctorate in Mathematics. She has a number of research publications in international journals and conferences of repute in her research area. Her research interests are numerical optimisation, nature inspired optimisation, computational intelligence, machine learning, artificial intelligence, especially genetic algorithms, parallel computing and particle swarm optimisation, etc.

Jagdish Chand Bansal is an Assistant Professor in Department of Applied Mathematics, South Asian University, India. He received his PhD in 2009. His research interests are particle swarm optimisation, memetic algorithms, parallel computing, artificial intelligence and their applications, etc.

This paper is a revised and expanded version of a paper entitled 'A non-deterministic adaptive inertia weight in PSO' presented at GECCO 2011 – 13th Annual Genetic and Evolutionary Computation Conference, Dublin, Ireland, 12–16 July 2011.

1 Introduction

Particle swarm optimisation (PSO) is a swarm-based global optimisation technique. It was introduced by Kennedy and Eberhart (1995) and developed by Kennedy et al. (2001). 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. Only within a few years of its introduction, PSO has gained wide popularity due to its simple mathematical operations, only a few parameters to adjust, ease of implementation and quick convergence.

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 on the experience of its best neighbour. 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) of the swarm. 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 and position for each particle is calculated according to the following velocity and position update equations:

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

In order to keep the particles within the search space the velocity in d^{th} dimension is restricted in the range $[-V_{d\max}, V_{d\max}]$ using the following relations (Eberhart et al., 1996):

$$\left.\begin{aligned} v_{id} &= -V_{d\max} \quad \text{if } v_{id} < -V_{d\max} \\ v_{id} &= V_{d\max} \quad \text{if } v_{id} > V_{d\max} \end{aligned}\right\} \tag{2}$$

The new position of the particle is then calculated, according to the following position update equation:

$$x_{id} = x_{id} + v_{id} \tag{3}$$

This process is then iterated until a predefined stopping criterion is satisfied. Here, $X_i = (x_{i1}, x_{i2}, \ldots, x_{id})$ represents the position of the i^{th} particle in a d-dimensional search space, $P_{besti} = (p_{i1}, p_{i2}, \ldots, p_{id})$ is i^{th} particle's pbest position, $P_{gbest} = (p_{g1}, p_{g2}, \ldots, p_{gd})$ is the gbest position and $V_i = (v_{i1}, v_{i2}, \ldots, v_{id})$ 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, by weighing the personal experience of a particle and the social information respectively. Low values allow particles to roam far from target regions before being tugged back, while high values result in abrupt movement towards, or past, target regions. Typically, these are both set equal to a value of 2.0. 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.

This model of PSO usually shows fast convergence rate during the earlier iterations and quickly approaches the solution region, but in the later iterations convergence rate decreases and it finds difficulty in refining the solution to a satisfactory level. The cause for this problem is hidden in the search process of PSO.

Actually, the search for global optimal solution in PSO 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 (Engelbercht, 2005). So an optimal balance between exploration and exploitation is the key to the performance control of PSO. To control this balance and the convergence behaviour of swarm the concept of inertia weight in PSO was introduced by Shi and Eberhart (1998a) as shown in the following equation:

$$v_{id} = \underbrace{wv_{id}}_{\substack{\text{Inertia} \\ \text{component}}} + \underbrace{c_1 r_1 \left(p_{id} - x_{id} \right)}_{\substack{\text{Cognitive} \\ \text{component}}} + \underbrace{c_2 r_2 \left(p_{gd} - x_{id} \right)}_{\substack{\text{Social} \\ \text{component}}} \tag{4}$$

where w is the inertia weight. Clearly, if c_1 and c_2 are kept fixed, bigger values of inertia weight will give large incremental changes in velocity per iteration which means more exploration. However, smaller inertia weight means less variation in velocity to provide slower update for fine tuning a solution, i.e., local search. It has been inferred that the system should start with a high inertia weight for more global exploration and then it should be decreased successively to facilitate finer local explorations. The value of inertia weight is generally taken between 0 and 1.

Since the introduction of inertia weight in PSO, a lot of research has been devoted to find its optimal or the standard value. But results show that its value is problem dependent and no standard value has yet been found. Initial implementations of the inertia weight used a static value for the entire process of evolution, for all particles and for each dimension. The problem with static inertia weight is that it is not capable of achieving optimal balance between local and global search. So 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. Therefore, it is particularly important to research dynamic inertia weight.

Many researchers have proposed a lot of strategies for dynamically adjusting inertia weight.

Shi and Eberhart (1998b) suggested the use of a linearly decreasing inertia weight starting from a large value (0.9) to a small value (0.4). They also proposed the use of random inertia weight in Shi and Eberhart (2001) and fuzzy-based adaptive inertia weight in Eberhart and Shi (2001). Zhan et al. (2009) also investigated decreasing inertia weight and found that the adaptive control of the inertia weight makes the algorithm extremely efficient, offering a substantially improved convergence speed in terms of both number of functional evaluations and CPU time needed to reach acceptable solutions. In contrast, Zheng et al. (2003a, 2003b) investigated increasing inertia weight. According to them, a PSO with increasing inertia weight outperforms the one with decreasing inertia weight both in convergence speed and solution precision, with no additional computing load. Analysing the advantages and disadvantages of above mentioned adaptive inertia weight strategies, Cui and Zhu (2007) found that with decreasing inertia weight, algorithm had good global search ability initially and had a better convergence later but with a slow rate while with increasing inertia weight, swarm converges faster initially but the local search ability was poor later. So they proposed a new method that was first linearly increasing inertia weight and then linearly decreasing it. A number of other ways for non-linear adaptation of inertia weight have been proposed in Chatterjee and Siarry (2006), Malik et al. (2007), Jiao et al. (2008), Dong et al. (2008), Li et al. (2009), Cheng and Wang (2011), and Sun et al. (2011). Liu et al. (2010) proposed a fuzzy weight strategy for PSO where the inertia weight reserves its decreasing property after fuzzy treatment, and the position is controlled by fuzzy parameter. Ahmad (2011) proposed an adaptive inertia weight approach which uses the success rate (SR) of the swarm as its feedback parameter to ascertain the particles' situation in the search space.

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. This paper investigates a non-deterministic way of dynamically adjusting the inertia weight proposed by Deep et al. (2011). This method is based on the improvement in the best fitness of the particles as the search process progresses. The performance of the PSO variants arising from it is compared with some other available PSO variants reported in the literature.

The remainder of this paper is organised as follows: Section 2 describes the investigated strategy for dynamic adjustment of inertia weight. Computational results obtained for the test functions are presented in Section 3 and concluded in Section 4. Finally, some limitations of the method with a direction for future work are given in Section 5.

2 Nature inspired adaptive inertia weight

This paper investigates a greedy approach for inertia weight in PSO that adapts itself at each iteration according to the improvement in the best fitness. This approach is inspired from human behaviour that "a success of one's act increases one's self-confidence, while a failure decreases it". Since the inertia weight in PSO is a measure of the particle's confidence in its present velocity. So the adaptation strategy is that the inertia weight should be increased when a better fit position is encountered and it should be decreased when a not so fit position is encountered. This modification, however, does not prevent

hill climbing capabilities of PSO, it merely increases the influence of potentially fruitful inertial directions while decreasing the influence of potentially unfavourable inertial directions. In this algorithm, the inertia weight has been taken as a function of iteration number and is updated according to the following equation:

$$\left.\begin{aligned} w(t+1) &= 0.9, && \text{if } t = 0 \\ w(t+1) &= f(t-1) - f(t) && \text{if } t > 0 \end{aligned}\right\} \tag{5}$$

where $w(t + 1)$ is the inertia weight at $(t + 1)^{\text{th}}$ iteration and $f(t)$ is the objective function value at t^{th} iteration. 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 $[-V_{d\max}, V_{d\max}]$ to keep the particles within the boundaries of the search space.

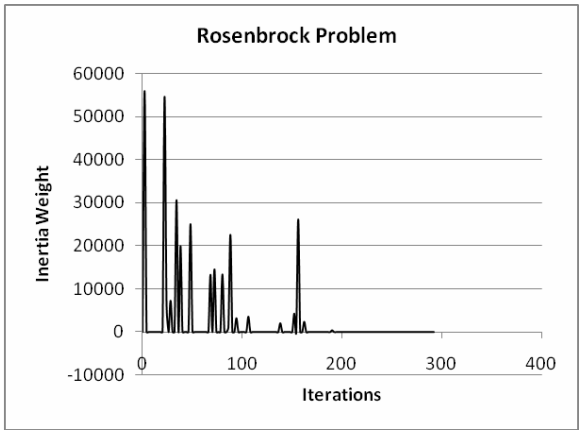
When the algorithm starts, employing larger inertia weight values, may lead to a strong exploration behaviour among swarms. On the other hand, decreasing inertia weight at the algorithm's final steps may direct the swarms to converge faster to global optimum which the algorithm has found yet. Hence, it could search for the better solution in the smaller region. When using this greedy approach, the variations of inertia weight as obtained by our experiments are shown (for three functions) in the Figure 1. Here, the value of inertia weight was updated using equation (5) according to the improvement in global best fitness. For all the functions used in this study, it was observed that there were 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 was observed that the oscillations become smaller and smaller which facilitate fine tuning of the final solution. Based on this observation, one 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 best position sometimes stops improving, i.e., 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, it should be provided with some inertia to increase the diversity. So the inertia weight equation is modified as follows:

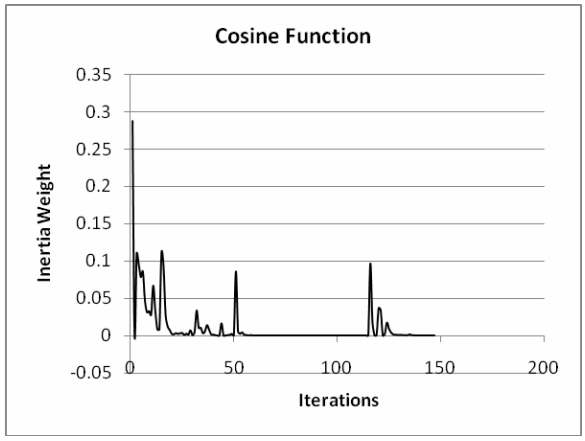
$$\left.\begin{aligned} & w(t+1) = 0.9 \\ & \text{and for } t > 0 \quad \text{we have} && \text{if } t = 0 \\ & w(t+1) = f(t-1) - f(t) \\ & w = w_{start} - (w_{start} - w_{end}) * t / t_{max}; && \text{if } w = 0 \text{ for } M \text{ successive iterations} \end{aligned}\right\} \tag{6}$$

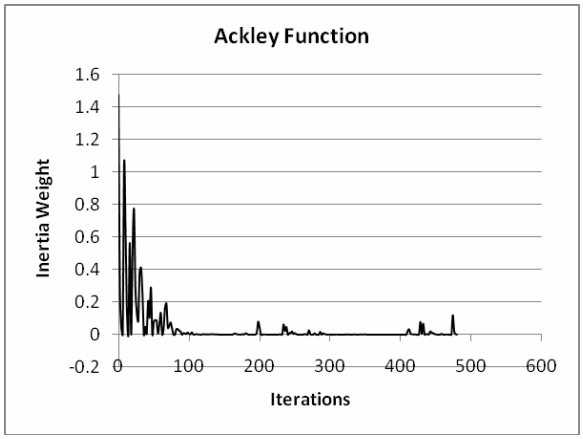
where w_{start} is the initial value and w_{end} is the final value of the inertia weight, and t is the current iteration (generation) of the algorithm while $t_{\max}$ is the maximum number of iterations (generations) specified by the user. This strategy may help in decreasing the period of entrapment (i.e., the number of iterations for which the swarm stagnates) in

suboptimal solutions during the search and hence improves the convergence rate. This strategy is used in two ways, namely, globally and locally.

Figure 1 Variation of inertia weight with iterations for Rosenbrock, Cosine and Ackley functions







In the global strategy each particle in the swarm has the same inertia weight that updates according to equations (6) using the improvement in global best fitness. It is called globally adaptive inertia weight (GAIW). The pseudo code for GAIW PSO is given in Algorithm 1. In this case, if the global best fitness improves at any iteration, 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. But, if the global best particle stops moving for a few iterations, 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 (6) using the improvement in its own personal best fitness. It is called locally adaptive inertia weight (LAIW). The pseudo code for LAIW PSO is given in Algorithm 2.

Algorithm 1 GAIW PSO

```
Begin
Initialize the swarm with random positions of particles.
Initialize the velocity of each particle to zero.
Evaluate the fitness of all particles.
Set current position of each particle as its pbest.
Set the best position of the swarm as gbest.
Initialize inertia weight at 0.9 and set iteration=0.
Do until termination condition is satisfied
iteration=iteration+1
For i=1 to swarm size
    For j=1 to dimension
        update velocity using equations (4) and (2)
        update position using equation (3)
    End For j
End For i
Evaluate fitness of all particles
update pbest and gbest
update inertia weight using equation (6)
End Do
End.
```

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 exploring globally. 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.

Algorithm 2 LAIW PSO

Begin
Initialize the swarm with random positions of particles.
Initialize the velocity of each particle to zero.
Evaluate the fitness of all particles.
Set current position of each particle as its pbest.
Set the best position of the swarm as gbest.
Initialize inertia weight at 0.9 and set iteration=0.
Do until termination condition is satisfied
iteration=iteration+1
For i=1 to swarm size
  For j=1 to dimension
    update velocity using equations (4) and (2)
    update position using equation (3)
  End For j
Evaluate fitness of ith particle
update inertia weight using equation (6)
End For i
update pbest and gbest
End Do
End.

The investigated 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 (7) and (8) are used to determine LDIW (Shi and Eberhart 1998b) and NDIW (Li et al., 2009), respectively.

$$w = w_{start} - \left(w_{start} - w_{end}\right) * t / \ t_{max} \tag{7}$$

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

where w_{start}, w_{end}, $t_{\max}$ and t have the same meanings as in equations (6), 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 w and t.

3 Computational experiments

This section focuses on comparing the effectiveness and performance of the investigated inertia versions GAIW and LAIW against FIW, LDIW and NDIW PSO as tested on 15 benchmark problems. All the PSOs are implemented in C and experiments are carried out on a Xeon, 3.4 GHz machine under LINUX operating system.

Table 1 Description of test function

Problem number	*Name*	*Function*	*Bounds*	ε	f_{min}
1	Sphere	$\sum_{i=1}^{n} x_i^2$	$[-5.12, 5.12]^{30}$	0.001	0
2	D Jong's F4-no noise	$\sum_{i=1}^{n} i x_i^4$	$[-5.12, 5.12]^{30}$	0.001	0
3	Griewank	$1+\frac{1}{4,000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right)$	$[-600, 600]^{30}$	0.001	0
4	Rosenbrock	$\sum_{i=1}^{n} \left(100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2\right)$	$[-30, 30]^{30}$	100	0
5	Rastrigin	$10n + \sum_{i=1}^{n} \left(x_i^2 - 10\cos\left(2\pi x_i\right)\right)$	$[-5.12, 5.12]^{30}$	50	0
6	Ackley	$-20\exp\left(-0.02\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n} \cos\left(2\pi x_i\right)\right) + 20 + e$	$[-30, 30]^{30}$	0.001	0
7	Levy Montalvo 1	$\frac{\pi}{n}\left(10\sin^2\left(\pi y_1\right) + \sum_{i=1}^{n-1}\left(y_i - 1\right)^2\left[1 + 10\sin^2\left(\pi y_{i+1}\right)\right] + \left(y_n - 1\right)^2\right)$ where $y_i = 1 + \frac{1}{4}\left(x_i + 1\right)$	$[-10, 10]^{30}$	0.001	0
8	Levy Montalvo 2	$0.1\left(\sin^2\left(3\pi x_1\right) + \sum_{i=1}^{n-1}\left(x_i - 1\right)^2\left[1 + 3\sin^2\left(3\pi x_{i+1}\right)\right] + \left(x_n - 1\right)^2\left[1 + \sin^2\left(2\pi x_n\right)\right]\right)$	$[-5, 5]^{30}$	0.001	0
9	Ellipsoidal	$\sum_{i=1}^{n} \left(x_i - i\right)^2$	$[-30, 30]^{30}$	50	0
10	Cosine	$\sum_{i=1}^{n} x_i^2 - 0.1\sum_{i=1}^{n} \cos\left(5\pi x_i\right)$	$[-1, 1]^{30}$	0.001	0
11	Exponential	$-\exp\left(-0.5\sum_{i=1}^{n} x_i^2\right)$	$[-1, 1]^{30}$	0.001	-1
12	Zakharov	$-\sum_{i=1}^{n} x_i + \left(\sum_{i=1}^{n} \frac{i}{2} x_i\right)^2 + \left(\sum_{i=1}^{n} \frac{i}{2} x_i\right)^4$	$[-5.12, 5.12]^{30}$	10	0
13	Cigar	$x_1^2 + 10,000\sum_{i=2}^{n} x_i^2$	$[-10, 10]^{30}$	50	0
14	Brown 3	$\sum_{i=1}^{n-1}\left\{\left(x_i^2\right)^{\left(x_{i+1}^2+1\right)} + \left(x_{i+1}^2\right)^{\left(x_i^2+1\right)}\right\}$	$[-1, 4]^{30}$	50	0
15	Schwefel 3	$\sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	$[-10, 10]^{30}$	0.001	0

3.1 Test problems

In this paper, a test bed of 15 benchmark problems with varying difficulty levels, is considered for evaluation and comparison of performance of the algorithm. These problems are scalable, i.e., the dimension of the problems can be increased/decreased as desired. In general, the complexity of the problem increases as the problem size is increased. The problem set is listed in Table 1. These problems are of continuous variables and have different degree of multimodality. All these problems are of minimisation type and the problem size (n) for all problems is kept fixed at 30.

3.2 Parameter selection

Here, we use $c_1 = c_2 = 2$, which has generally been used in literature. The swarm size has been set as 60. The maximum allowable velocity in each dimension has been taken to be $V_{d\max} = (x_{d\max} - x_{d\min})$ *0.25, i.e., under the assumption that any particle at any iteration can have a sufficient excursion without escaping the optimum in a given direction, where $x_{d\max}$ and $x_{d\min}$ are the upper and lower bounds for particles' positions in d^{th} dimension of the search space. In order to make fair comparisons, all these settings have been kept same for the compared algorithms for a given problem. The maximum number of iterations $t_{\max}$ is fixed as 5,000 for all algorithms and for all problems. The termination criterion for all algorithms is a combination of the following two conditions:

1 reaching the maximum number of iterations

2 getting the minimum error ε (i.e., getting a solution within the tolerance limit ε), 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 $w_{end} = 0.4$ (Shi and Eberhart, 1998b) are set. Also for NDIW PSO $k = 0.6$ [as recommended in Li et al. (2009)] is taken. For GAIW and LAIW, $M = 25$ is taken for all scalable 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, i.e., for each of the 100 runs a different seed is used for starting the random number generator. All the results that are 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 (AFE) of successful runs.

2 $$\text{SR} = \frac{\text{number of successful runs}}{\text{total number of runs}} \times 100$$

3 Success performance (SP) $= \dfrac{\text{AFE}}{\text{number of successful runs}} \times \text{total number of runs}$

(Liang et al., 2006)

4 Average execution time (AET) of successful runs.

5 Average error (AE) = Average of $|f_{\min} - f_{opt}|$ over successful runs.

6 Standard deviation (SD) = SD of the error $|f_{\min} - f_{opt}|$.

These performance metrics are calculated for each problem-algorithm pair and are shown in tabular as well as graphical form.

3.4 Results and discussions

From the recorded simulated results, statistical analyses are carried out and presented in Tables 2 to 7 and Figures 2 to 8. In Figures 2 to 7, the best performing PSO is marked with star. For analysis box-plots have been used. These have been drawn for all the functions of the problem set, taking one problem set at a time. For drawing box-plots, the problems with no success are assigned maximum number of function evaluations and the maximum SP.

Table 2 Average number of functional evaluations

Problem number	*Average number of functional evaluations*				
	FIW	*LDIW*	*NDIW*	*GAIW*	*LAIW*
1	70,921	161,621	99,573	15,080	1,560
2	72,207	159,547	97,753	21,789	1,645
3	157,472	187,242	124,739	36,272	1,867
4	121,825	186,560	120,594	140,243	1,480
5	64,856	180,404	96,820	45,017	651
6	177,488	194,984	131,728	72,143	119,713
7	38,878	142,139	81,024	9,536	31,769
8	60,750	156,436	93,478	19,351	192
9	23,821	-	51,660	14,590	-
10	69,359	157,254	96,107	22,424	7,463
11	33,976	139,144	79,050	8,045	10,393
12	26,763	126,875	61,943	23,941	480
13	86,864	167,530	105,958	195,658	1,560
14	421	543	1,250	1,242	45,937
15	177,488	180,559	119,416	42,372	1,980

The goal of the analysis is to observe if the proposed strategy shows an improvement over the existing ones or not. Table 2 shows the AFE of successful runs. Corresponding box-plots are given in Figure 2 to observe the performances of PSOs at a glance. AFE is proportional to the computational cost of the method. It is clear that LAIW performs the best and GAIW stands second from AFE point of view. The order of PSOs based on the computational cost is given below:

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

It can be clearly seen that the proposed PSOs significantly reduce the AFE and hence the computational effort.

Table 3 Success rate

Problem number	*Success rate*				
	FIW	*LDIW*	*NDIW*	*GAIW*	*LAIW*
1	100	100	100	100	100
2	98	90	100	100	100
3	21	31	39	17	100
4	74	53	75	94	100
5	81	68	94	98	100
6	97	97	100	99	64
7	71	53	96	83	100
8	96	99	96	95	100
9	4	0	1	97	0
10	97	98	100	97	100
11	100	99	100	99	99
12	100	95	99	100	100
13	39	20	37	92	100
14	100	100	99	100	51
15	81	55	91	15	100

Table 4 Success performance

Problem number	*Success performance*				
	FIW	*LDIW*	*NDIW*	*GAIW*	*LAIW*
1	70,921	161,621	99,573	15,080	1,560
2	73,680.6	177,274.4	97,753	21,789	1,645
3	749,866.7	604,006.5	319,843.6	213,364.7	1,867
4	164,628.4	352,000	160,792	149,194.7	1,480
5	80,069.1	265,300	103,000	45,935.71	651
6	182,977.3	201,014.4	131,728	72,871.72	187,051.6
7	54,757.8	268,186.8	84,400	11,489.16	31,769
8	63,281.3	158,016.2	97,372.92	20,369.47	192
9	595,525	-	5,166,000	15,041.24	-
10	71,504.1	160,463.3	96,107	23,117.53	7,463
11	33,976	140,549.5	79,050	8,126.263	10,497.98
12	26,763	133,552.6	62,568.69	23,941	480
13	222,728.2	837,650	286,373	212,671.7	1,560
14	421	543	1,262.626	1,242	90,072.55
15	219,121	328,289.1	131,226.4	282,480	1,980

Table 5 Average execution time

Problem number	*Average execution time*				
	FIW	*LDIW*	*NDIW*	*GAIW*	*LAIW*
1	0.303199	0.668365	0.502401	0.077768	0.008834
2	0.308057	0.667041	0.497823	0.112661	0.009385
3	1.096168	1.302863	0.982663	0.290529	0.015888
4	0.617346	0.956865	0.728171	0.848851	0.009879
5	0.598862	1.091694	0.672774	0.314717	0.00475
6	1.156096	1.24611	0.962913	0.536842	0.941187
7	0.364262	1.149043	0.730205	0.088965	0.304798
8	0.536564	1.209795	0.808961	0.173119	0.001594
9	0.151256	-	0.38075	0.108605	-
10	0.445497	0.969521	0.680739	0.163085	0.059022
11	0.160474	0.596439	0.410468	0.042905	0.05556
12	0.208266	0.589197	0.345083	0.133899	0.002781
13	0.384197	0.688723	0.531786	0.982588	0.008784
14	0.006215	0.007833	0.020361	0.020535	0.671959
15	0.678231	0.960435	0.743937	0.270227	0.013654

Table 6 Average error

Problem number	*Average error*				
	FIW	*LDIW*	*NDIW*	*GAIW*	*LAIW*
1	0.000953	0.000945	0.000942	0.000934	0.000114
2	0.000935	0.000895	0.000919	0.000945	0
3	0.000954	0.000924	0.000925	0.000885	0.000007
4	98.34058	97.89938	98.88599	94.54052	32.66543
5	49.21176	49.39174	49.19422	49.4536	4.322524
6	0.000971	0.000966	0.000965	0.000928	0
7	0.000947	0.000954	0.000935	0.000912	0.93914
8	0.000936	0.000938	0.00093	0.000881	7.658117
9	48.0878	-	49.38712	47.7859	-
10	0.000938	0.000947	0.000935	0.000875	0.00024
11	0.000942	0.00094	0.000948	0.000921	0.000937
12	9.348343	9.480324	9.200464	9.336067	0
13	46.79	47.62784	47.17672	44.49966	6.322957
14	32.82387	45.37833	42.70737	44.3987	44.94361
15	0.000948	0.000954	0.000964	0.000799	0

Table 7 Standard deviation

Problem number	*Standard deviation*				
	FIW	*LDIW*	*NDIW*	*GAIW*	*LAIW*
1	0.000034	0.000057	0.000068	0.000065	0.000126
2	0.00016	0.000316	0.000079	0.00006	0
3	0.001852	0.001381	0.001158	0.00196	0.000067
4	58.35779	92.35525	57.10645	28.01283	11.49765
5	23.86082	33.89431	12.4941	7.110318	7.890891
6	0.000173	0.000173	0.000029	0.00015	0
7	0.000607	0.000899	0.000199	0.000438	0.053908
8	0.00021	0.000116	0.000204	0.00024	1.539504
9	235.5884	-	491.3956	8.68543	-
10	0.000179	0.000145	0.00006	0.000208	0.000406
11	0.000056	0.000112	0.000053	0.000102	0.000113
12	0.669744	2.24042	1.179013	0.56194	0
13	58.60587	95.27074	61.61783	17.83137	3.943056
14	0	0.705244	6.800679	5.472719	44.26714
15	0.000462	0.000863	0.000305	0.001913	0

Figure 2 Average number of functional evaluations (see online version for colours)

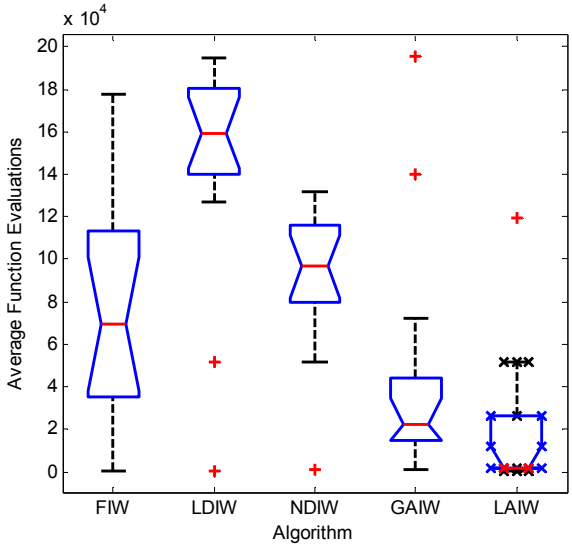


Table 3 compares SR of all five versions of the PSO considered in this paper. SR is directly proportional to the reliability of the method. The corresponding boxplot is shown

in Figure 3. It is clear that performance of LAIW is again the best among all PSOs considered. i.e., proposed local strategy shows an improvement over global one and other PSOs. Further, the order of PSOs based on the SR is as follows:

LAIW > GAIW > NDIW > FIW > LDIW.

Though the SR of LAIW for the problem number 9 is zero, GAIW showed a very good SR for this problem while that for others is very small. It may be seen as an exceptional case for LAIW.

Figure 3 Success rate (see online version for colours)

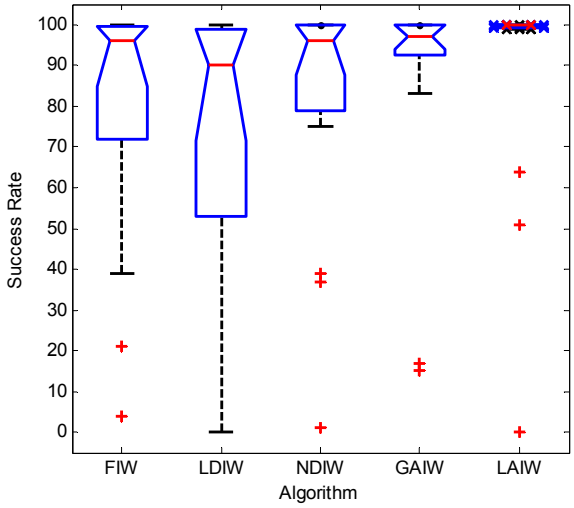


Figure 4 Success performance (see online version for colours)

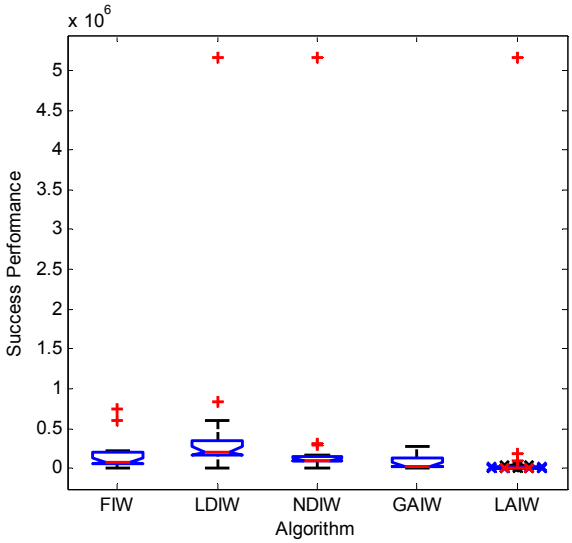


Figure 5 Average execution time (see online version for colours)

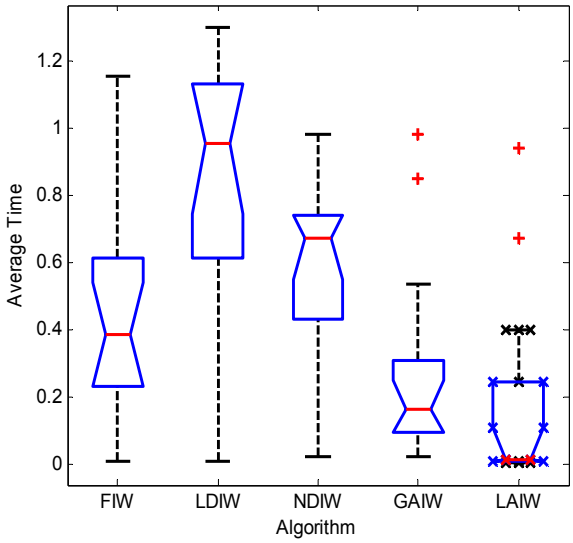
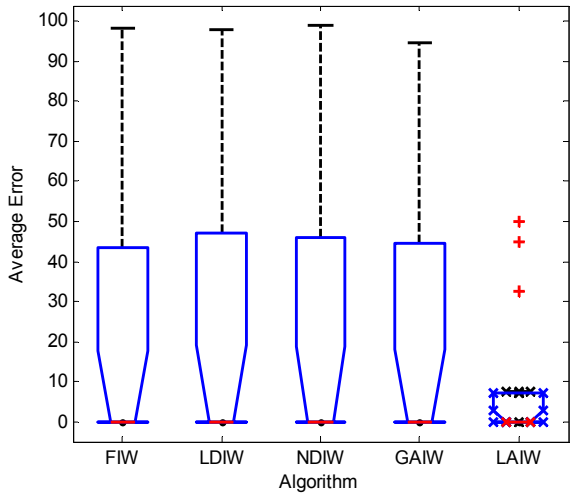


Figure 6 Average error (see online version for colours)



In order to observe the consolidated effect on SR and AFE on the performance of PSO, a comparison among all five versions is made on the basis of SP also. Table 4 and Figure 4 present this information. From SP point of view following order is seen:

LAIW > GAIW > NDIW > FIW > LDIW.

The AET of the successful runs of all considered PSOs is presented in Table 5 and corresponding box-plot is shown in Figure 5. Clearly, LAIW is again the best performer. The order of PSOs based on the AET is given below:

LAIW > GAIW > FIW > NDIW > LDIW.

Also, it can be seen that by using the proposed strategy the AET decreases significantly, i.e., the convergence rate is considerably improved.

Figure 7 Standard deviation (see online version for colours)

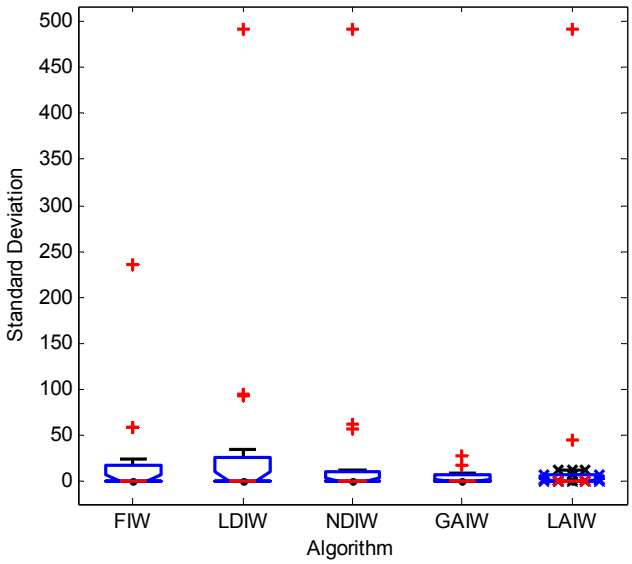
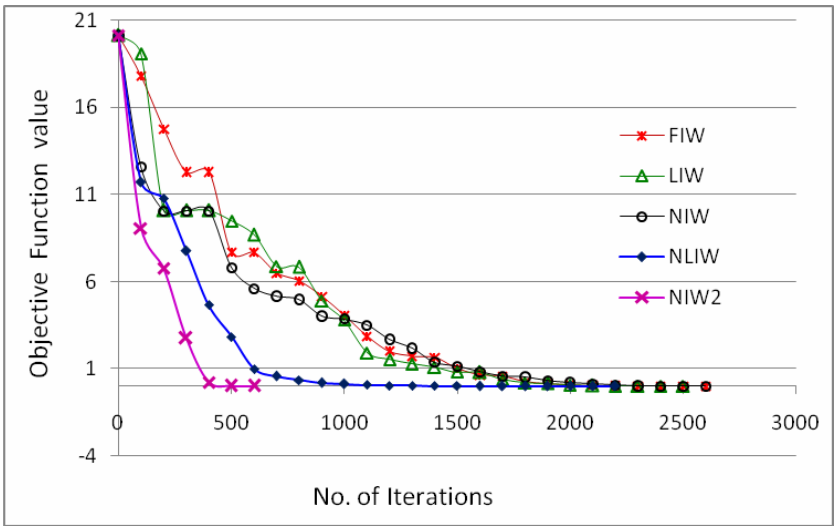


Figure 8 Convergence graph for sphere function (see online version for colours)



.

Now the most accurate method is sought. For this, the comparisons based on AE and SD of successful runs are carried out. SD gives the information about the consistency of the algorithm to reach the optimal solution over the successful runs. Smaller value of SD indicates the consistency of the algorithm in finding the optimal solution. Tables 6 and 7 show AE and SD respectively and corresponding box-plots are given in Figures 6 and 7, respectively. Due to the termination condition based on error, there is little difference among AE of all five versions of the algorithms and the same is true for SD. From Figure 5, it is clear that on the basis of AE, LAIW performs the best. All other versions are comparable. From Figure 7, it is observed that on the basis of SD also, LAIW is the best performer. Following performance order of the algorithms, for solution accuracy, is observed:

LAIW > GAIW > FIW > NDIW > LDIW.

On the basis of above analysis, LAIW performs the best among all five versions of PSO. Although in an exceptional case when LAIW could not solve the problem number 9, GAIW performed the best among all others. So, it may be said that the proposed strategy gives significantly better results than the existing ones.

The convergence graph for sphere function for all the five versions of PSO is shown in the Figure 8. It clearly shows that LAIW gives the fastest convergence towards the minimum and can find the optimal solution. Thus, from the point of view of convergence rate also, proposed strategy performs the best.

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 affects, almost equally, the movement of the particles during entire search process. But in the proposed strategy when the value of w is very large, the particles perform almost individual (i.e., independently of others) 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 optimisation algorithm. By studying the patterns of variation of inertia weight for various functions used here, it may be observed that during the initial iterations, w takes very large values, so initially individual search is more effective than social search. 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 $w = 0.9$ which first explore the search space vigorously, sometimes searching locally around good positions and in the later iterations concentrates the search in good areas found so far. Hence, it can be said that the faster convergence of proposed PSO is due to:

1 the quick exploration in first few iterations

2 the reduction in the period of entrapment during the search.

Also the high SR may be due to the increased diversity of the swarm because now the swarm explores the search space more thoroughly.

4 Conclusions

In this paper, two inertia weight variants of PSO namely GAIW and LAIW PSO, which are based on a nature inspired approach for dynamically adjusting the inertia weight, have been investigated. These variants have been tested on 15 scalable problems and results thus obtained, have been compared with those obtained by three inertia weight variants taken from literature. With the LAIW PSO performing the best and GAIW performing the second best among the considered PSO variants, the results show that the proposed variants improve the performance of PSO significantly in terms of convergence speed as well as computational effort, and hence can be used for different kinds of optimisation problems, thus releasing the user from the pain of indulging into extensive experiments for finding an appropriate setting of inertia weight. We do not claim that the proposed strategy, for adaptation of inertia weight, is best for any problem in general, but it is recommended since it is found to be repeatedly giving good results for most of the problems studied here and hence it is an appropriate strategy to be used if we talk about the overall performance.

5 Limitations and scope for future work

As every coin has two faces, our algorithms also have some good aspects as shown above as well as some limitations: first is that it increases the number of control parameters by using w_{start}, w_{end} and M for just one parameter w, and the second, that was found during later of our study, is that for some functions the value of inertia weight becomes very large for many iterations and the swarm goes on searching on the boundary during those iterations. Another limitation of this strategy is that due to the explicit use of fitness function value f, the behaviour of PSO does no longer remain the same for equivalent functions (landscapes).

Also, the global topology for swarm has been used here, the effect of the proposed inertia weight strategies on the PSO with local topology may be somewhat different. Although these strategies give very good results for the benchmark problems used in this paper, its behaviour on various other benchmarks and real life optimisation problems is a matter of further investigation.

References

Ahmad, N., Mohammad, M.E. and Reza, S. (2011) 'A novel particle swarm optimization algorithm with adaptive inertia weight', *Applied Soft Computing*, Vol. 11, No. 4, pp.3658–3670.

Chatterjee, A. and Siarry, P. (2006) 'Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization', *Comput. Oper. Res*., Vol. 33, No. 3, pp.859–871.

Cheng, H.X. and Wang, X. (2011) 'An improved particle swarm optimization algorithm', *Adv. Mater. Res*., Vol. 186, pp.454–458.

Cui, H.M. and Zhu, Q.B. (2007) 'Convergence analysis and parameter selection in particle swarm optimization', *Comput. Eng. Appl*., Vol. 43, No. 23, pp.89–91.

Deep, K., Arya, M. and Bansal, J.C. (2011) 'A non-deterministic adaptive inertia weight in PSO', in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'11), ACM*, p.1155.

Dong, C., Wang, G., Chen, Z. et al. (2008) 'A method of self-adaptive inertia weight for PSO', in *Proceedings of 2008 International Conference on Computer Science and Software Engineering*.

Eberhart, R.C. and Shi, Y. (2001) 'Tracking and optimizing dynamic systems with particle swarms', in *Proceeding Congress on Evolutionary Computation 2001*, IEEE Service Centre, Seoul, Korea, Piscataway, NJ.

Eberhart, R.C., Simpson, P.K. and Dobbins, R.W. (1996) *Computational Intelligence PC Tools*, 1st ed., Academic Press Professionals, San Diego, CA, USA.

Engelbercht, A.P. (2005) *Fundamentals of Computational Swarm Intelligence*, John Wiley & Sons.

Jiao, B., Lian, Z. and Gu, X. (2008) 'A dynamic inertia weight particle swarm optimization algorithm', *Chaos, Solitons and Fractals*, Vol. 37, No. 3, pp.698–705.

Kennedy, J. and Eberhart, R.C. (1995) 'Particle swarm optimization', in *Proceedings of IEEE International Conference on Neural Networks*, Australia, WA.

Kennedy, J., Eberhart, R.C. and Shi, Y. (2001) *Swarm Intelligence*, Morgan Kaufmann Publishers, San Francisco.

Li, L., Xue, B., Niu, B. et al. (2009) 'A novel particle swarm optimization with non-linear inertia weight based on tangent function', in Huang, D.S. et al. (Eds.): *ICIC 2009. Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence)*, Vol. 5755, Springer-Verlag, Berlin, Heidelberg.

Liang, J., Runarsson, T., Mezura-Montes, E. et al. (2006) 'Problem definitions and evaluation criteria for the CEC 2006, special session on constrained real-parameter optimization', Technical Report.

Liu, C., Ouyang, C., Zhu, P. et al. (2010) 'An adaptive fuzzy weight PSO algorithm', *2010 Fourth International Conference on Genetic and Evolutionary Computing (ICGEC)*.

Malik, R.F., Rahman, T.A., Hashim, S.Z.M. et al. (2007) 'New particle swarm optimizer with sigmoid increasing inertia weight', *Int. J. Comput. Sci. Sec*., Vol. 1, No. 2, pp.35–44.

Shi, Y. and Eberhart, R.C. (1998a) 'A modified particle swarm optimizer', in *Proceedings of the IEEE Conference on Evolutionary Computation*, IEEE Press, Piscataway, NJ.

Shi, Y. and Eberhart, R.C. (1998b) 'Parameter selection in particle swarm optimization', *Lecture Notes in Computer Science*, Vol. 1447, Springer-Verlag, Berlin, Heidelberg, p.591.

Shi, Y. and Eberhart, R.C. (2001) 'Fuzzy adaptive particle swarm optimization', in *Proceedings of Congress on Evolutionary Computation 2001*, IEEE Service Centre, Seoul, Korea, Piscataway, NJ.

Sun, Y., Zhu, S., Li, Q. et al. (2011) 'A kind of decay-curve inertia weight particle swarm optimization algorithm', in Chen, R. (Ed.): *ICICIS 2011, Part I. CCIS*, Vol. 134, p.402, Springer-Verlag, Berlin, Heidelberg.

Zhan, Z.H., Zhang, J., Li, Y. et al. (2009) 'Adaptive particle swarm optimization', *IEEE Trans Systems, Man and Cybernetics-Part B: Cybernetics*, Vol. 39, No. 6, pp.1362–1381.

Zheng, Y.L., Ma, L.H., Zhang, L.Y. et al. (2003a) 'Empirical study of particle swarm optimizer with an increasing inertia weight', in *Proceeding of the IEEE Congress on Evolutionary Computation 2003*.

Zheng, Y.L., Ma, L.H., Zhang, L.Y. et al. (2003b) 'On the convergence analysis and parameter selection in particle swarm optimization', in Xi'an (Ed.): *Proceedings of the Second International Conference on Machine Learning and Cybernetics, 2003*.

Notes

1 A > B implies that algorithm A performs better than algorithm B in terms of the particular performance metric under consideration.