



Journal of Experimental & Theoretical Artificial Intelligence

ISSN: 0952-813X (Print) 1362-3079 (Online) Journal homepage: <http://www.tandfonline.com/loi/teta20>

An efficient oscillating inertia weight of particle swarm optimisation for tracking optima in dynamic environments

Javidan Kazemi Kordestani, Alireza Rezvanian & Mohammad Reza Meybodi

To cite this article: Javidan Kazemi Kordestani, Alireza Rezvanian & Mohammad Reza Meybodi (2016) An efficient oscillating inertia weight of particle swarm optimisation for tracking optima in dynamic environments, *Journal of Experimental & Theoretical Artificial Intelligence*, 28:1-2, 137-149, DOI: [10.1080/0952813X.2015.1020521](https://doi.org/10.1080/0952813X.2015.1020521)

To link to this article: <http://dx.doi.org/10.1080/0952813X.2015.1020521>



Published online: 27 Mar 2015.



Submit your article to this journal



Article views: 14



View related articles



View Crossmark data

Full Terms & Conditions of access and use can be found at
<http://www.tandfonline.com/action/journalInformation?journalCode=teta20>

An efficient oscillating inertia weight of particle swarm optimisation for tracking optima in dynamic environments

Javidan Kazemi Kordestani^a, Alireza Rezvanian^{b,c*} and Mohammad Reza Meybodi^b

^aDepartment of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran;

^bSoft Computing Laboratory, Computer Engineering and Information Technology Department, Amirkabir University of Technology (Tehran Polytechnic), 424 Hafez Ave., Tehran, Iran; ^cDepartment of Computer Engineering, Hamedan Branch, Islamic Azad University, Hamedan, Iran

(Received 18 March 2014; accepted 1 September 2014)

One of the effective techniques for improving the rate of convergence in the particle swarm optimisation (PSO) is modifying the inertia weight parameter. This parameter can specify the search area of the swarm in the environment and establish a good balance between the global and local search ability of the particles. Several strategies have been already suggested and well tested for setting the inertia weight in static environments. However, in dynamic environments, the effect of this parameter on increasing the ability of PSO in tracking the changing optimum has been barely considered. In this paper, a time-varying inertia weight, called oscillating triangular inertia weight, is presented and its performance is measured on the moving peaks benchmark (MPB). Experimental results on various dynamic scenarios generated by MPB demonstrate that the proposed strategy has a better capability to adapt with the environmental changes in comparison with other techniques including constant inertia weight and linearly decreasing inertia weight.

Keywords: particle swarm optimisation; inertia weight; oscillating triangular inertia weight; dynamic environments; moving peaks benchmark

1. Introduction

Many real-world optimisation problems are dynamic in which the optimum solution(s) to the problem may change over time. Examples of such problems which are known as dynamic optimisation problems (DOPs) include: dynamic function optimisation (Rezvanian & Meybodi, 2010a; 2010b), dynamic multicast problems in mobile ad hoc networks (Cheng & Yang, 2010), dynamic intrusion detection (Wei, Hou, Tan, & Guo, 2010) and dynamic job shop scheduling (Zandieh & Adibi, 2010). Several techniques based on evolutionary algorithms (EAs) have been presented over the years for solving DOPs, among them particle swarm optimisation (PSO) have attracted a great deal of attention (Conforth & Meng, 2010; Gogna & Tayal, 2013; Nickabadi, Ebadzadeh, & Safabakhsh, 2012). PSO is a population-based optimisation method which was first proposed by Kennedy and Eberhart (1995). The PSO, similar to the other algorithms belonging to the EAs family, is a stochastic algorithm that does not require gradient information to find an optimum (van den Bergh, 2002). This feature makes PSO suitable for functions where the gradient is either unavailable or computationally expensive. Moreover, PSO is easy to implement, has a high efficiency (Shi & Eberhart, 1998), and can be easily applied to a wide

*Corresponding author. Email: a.rezvanian@aut.ac.ir, rezvan@iauh.ac.ir

range of applications (Aghdam, Mirzaee, Pourmahmood, & Aghababa, [in press](#); Conforth & Meng, [2010](#); Liu, Yang, & Wang, [2010](#); Nabizadeh, Faez, Tavassoli, & Rezvanian, [2010](#); Nabizadeh, Rezvanian, & Meybodi, [2012](#); Nickabadi et al., [2012](#); Norouzzadeh, Ahmadzadeh, & Palhang, [2012](#); Rezaee Jordehi & Jasni, [2013](#); Soleimani-Pouri et al., [2012](#); Yazdani, Nasiri, Sepas-Moghaddam, & Meybodi, [2013](#)). There exist various studies that have combined good characteristics of PSO with other optimisation techniques (Gogna & Tayal, [2013](#)).

PSO contains a population of particles exploring in a D -dimensional search space with the aim to find promising regions. Each particle i has three features: \vec{x}_i that shows the current position of the particle i in the search space, \vec{v}_i which is the velocity of the particle i and \vec{p}_i which represents the best position found so far by the particle i . Furthermore, each particle can interact with its neighbours according to a population topology. There are two different topologies for communication between the particles of the swarm which are known as *gbest* (Kennedy & Eberhart, [1995](#)) and *lbest* (Ghosh, Das, Kundu, Suresh, & Abraham, [2012](#)). In *gbest* model, particles of the swarm update their velocity and position according to the following equations:

$$\vec{v}_i(t+1) = \vec{v}_i(t) + c_1 r_1 [\vec{p}_i(t) - \vec{x}_i(t)] + c_2 r_2 [\vec{p}_g(t) - \vec{x}_i(t)], \quad (1)$$

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1), \quad (2)$$

where c_1 and c_2 are acceleration coefficients (learning factors) which are used to determine the degree of the movement of particles towards their personal best position and global best position of the swarm, respectively. r_1 and r_2 are two independent random variables drawn with uniform probability from $[0,1]$. Finally, \vec{p}_g is the globally best position found so far by all the particles in the population.

Several modifications have been proposed for improving the performance of PSO (Hasanzadeh, Meybodi, & Ebadzadeh, [2013](#); Hasanzadeh, Meybodi, & Ghidary, [2011](#); Soleimani-Pouri, Rezvanian, & Meybodi, [2014](#)). One of the most widely used improvements is the introduction of the inertia weight parameter by *Shi* and *Eberhart* (Shi & Eberhart, [1998](#)). The inertia weight is used to regulate a balance between the exploration and exploitation capabilities of PSO. Regarding the inertia weight, the velocity update equation is as follows:

$$\vec{v}_i(t+1) = w\vec{v}_i(t) + c_1 r_1 [\vec{p}_i(t) - \vec{x}_i(t)] + c_2 r_2 [\vec{p}_g(t) - \vec{x}_i(t)]. \quad (3)$$

In which the original velocity update equation is obtained by setting $w = 1$. While larger values of inertia weight allows PSO to explore a wide area of the search space, smaller values of inertia weight compels particles to exploit around a confined region.

Numerous strategies have been introduced for adjusting the inertia weight parameter in static environments. For example, Hashemi and Meybodi ([2011](#)) introduced two classes of learning automata based algorithms for adaptive selection of value for inertia weight and acceleration coefficients in PSO. In both classes, they used a learning automaton per w , c_1 and c_2 in order to choose an optimal value for corresponding parameter at each stage of the algorithm. Nickabadi, Ebadzadeh, and Safabakhsh ([2011](#)) proposed a novel adaptive inertia weight which uses the *success rate* of the swarm as its feedback parameter to determine the situation of the particles in the search space. A comprehensive review on the topic of inertia weight strategies can be found in Nickabadi et al. ([2011](#)). Although inertia weight has been widely applied to static environments, the role of this parameter in dynamic environments has been barely studied. In this paper we present a new time-varying inertia weight, called *oscillating triangular inertia weight*, for dynamic environments. Experimental results show that the proposed inertia weight can improve the performance of the PSO in dynamic environments. The rest of this paper is

organised as follows: Section 2 provides a brief review of different inertia weight strategies in dynamic environments. Proposed strategy is then introduced in Section 3. Experimental setup is presented in Section 4. Section 5 is devoted to the experimental results and analysis. Finally, Section 6 concludes the paper with discussions and future works.

2. Inertia weight in dynamic environments

Since the inclusion of inertia weight parameter to the PSO by Shi and Eberhart (1998), several strategies have been proposed to adjust the value of this parameter. Nickabadi et al. (2011) classified different inertia weight adjustment strategies into three classes:

- (1) Constant or random inertia weight in which the value of the inertia weight is either constant during the search process or is determined randomly (Shi & Eberhart, 1998).
- (2) Time-varying inertia weight where the inertia weight is defined as a function of time or iteration number (Shi & Eberhart, 1998).
- (3) Adaptive inertia weight that includes those strategies which use a feedback parameter to monitor the state of the algorithm, and adjust the value of the inertia weight accordingly (Feng, Cong, & Feng, 2007).

In the rest of this section, we will briefly review different PSO algorithms for dynamic environments in terms of their inertia weight adjustment strategies.

2.1 Constant inertia weight

The majority of the PSO-based methods for DOPs have utilised a constant inertia weight in their velocity update equation. For instance, Kamosi, Hashemi, and Meybodi (2010a, 2010b) introduced a multi-swarm algorithm for dynamic environments, referred to as mPSO, which consists of a parent swarm to explore the search space and a varying number of child swarms to exploit the promising areas found by the parent swarm. In another work, inspired by hibernation phenomenon in animals, they extended mPSO and proposed a hibernating multi-swarm algorithm (Kamosi, Hashemi, & Meybodi, 2010a, 2010b), named HmSO. In HmSO, unproductive search in child swarms is stopped by ‘sleeping’ converged child swarms. Hashemi and Meybodi (2009a, 2009b) incorporated PSO and cellular automata into an algorithm referred to as cellular PSO. In cellular PSO, the search space is partitioned into some equally sized cells using cellular automata. Then, particles of the swarm are allocated to different cells according to their positions in the search space. At any time, particles residing in each cell use their best personal experience and the best solution found in the neighbourhood of the cell for searching an optimum. In Hashemi and Meybodi (2009a, 2009b), they changed the role of some particles in each cell, from standard particles to quantum particles for a few iterations upon the detection of a change in the environment.

2.2 Time-varying inertia weight

An example of time-varying inertia weight strategy in dynamic environments is the clustering particle swarm optimiser (CPSO) by Yang and Li (2010). In CPSO, a hierarchical clustering method is employed to create an adaptive number of sub-swarms and assign each of them to different promising areas of the search space. Each created sub-swarm is then responsible for exploring its respective sub-region. In order to speed up the convergence of the sub-swarms, they applied a linearly decreasing strategy in the range of $[w_{\min}, w_{\max}] = [0.3, 0.6]$.

2.3 Adaptive inertia weight

In this class of strategies a feedback parameter, which is received from the environment (or the algorithm), is used to adjust the value of the inertia weight at each stage of the optimisation process. For example, Nickabadi et al. (2012) proposed a competitive clustering particle swarm optimiser (CCPSO) for DOPs. They employed a multi-stage clustering procedure to split the particles of the main swarm over a varying number of sub-swarms based on the particles positions as well as their objective function values. Moreover, each sub-swarm adjusts its inertia weight according to the *success rate* of the sub-swarm (Nickabadi et al., 2011).

3. Proposed strategy

Oscillating inertia weight first proposed and studied in static environments by Kentzoglannakis and Poole (2009). The main idea of this strategy is to trigger a wave of global search (explore) followed by a wave of local search (exploit), forming a cycle which is repeated during the optimisation process. In this paper, the fundamental idea of oscillating inertia weight is applied to dynamic environments. A very basic symmetric triangular wave function is used to determine the value of inertia weight at each time step. This function hybridises the linearly decreasing inertia weight and the linearly increasing inertia weight into a single inertia weight called oscillating triangular inertia weight. The oscillating triangular inertia weight is expressed by the following equation:

$$w_{i,t} = w_{\max} - |\text{round}(\alpha t) - \alpha t|, \quad (4)$$

where $w_{i,t}$ is the value of inertia weight at t th iteration after i th change; t is the iteration counter which starts from zero and increases by one at the end of each iteration, until the environment changes where t is set to zero again. w_{\max} is the maximum value of the inertia weight. $\alpha \in [0.0, 0.5]$ is used to generate different shaped waveforms for oscillating triangular inertia weight. This parameter can control the values of oscillating triangular inertia weight in the range of [0.4, 0.9]. Finally, $\text{round}(x)$ is the function that rounds off real number x to its nearest integer. Figure 1 shows the values of oscillating inertia weight over time for $\alpha = 0.1$ and $\alpha = 0.3$.

The outline of the PSO with oscillating triangular inertia weight is shown in Figure 2.

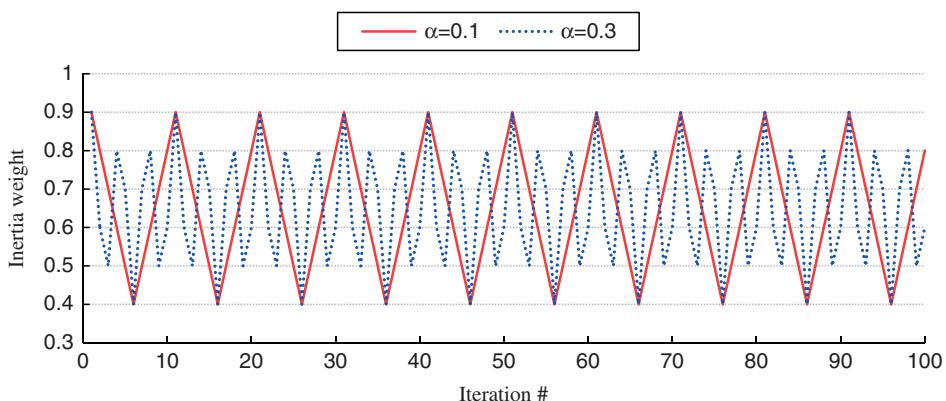


Figure 1. Inertia weight values in oscillating triangular inertia weight PSO during 100 iterations for $\alpha = 0.1$ and $\alpha = 0.3$.

Algorithm 1. Oscillating triangular inertia weight PSO for dynamic environment

```

1. Setting parameters
2. for each particle  $i$  in the swarm do
3.   |Randomly initialize  $\vec{v}_i, \vec{x}_i = \vec{p}_i$ 
4. end-for
5. evaluate the population
6. set  $\vec{p}_g$  to the best position in the search space found by the particles so far
7. While final condition not met do
8.   |Update inertia weight according to Eq. (4)
9.   |if a change is detected in the environment then
10.    |  |re-evaluate memory of all particles
11.    |  |set  $\vec{p}_g$  to the best position in the search space found by the particles so far
12.    |  |t=0;
13.   |end-if
14.   |for each particle  $i$  in the population do
15.     |  |Update position of  $i$  according to Eq. (3) and Eq. (2), respectively.
16.     |  |if(fitness( $\vec{p}_i$ )<fitness( $\vec{x}_i$ )) then
17.       |  |  | $\vec{p}_i = \vec{x}_i$ 
18.       |  |  |if(fitness( $\vec{p}_g$ )<fitness( $\vec{x}_i$ )) then
19.         |  |  |  | $\vec{p}_g = \vec{x}_i$ 
20.         |  |  |end-if
21.       |  |end-if
22.     |end-for
23.   |t = t + 1;
24. end while

```

Figure 2. Pseudo code of oscillating triangular inertia weight PSO.

In this algorithm after initialising the swarm, particles start to explore the search space. At the beginning of each iteration, the inertia weight parameter is updated according to [Equation \(4\)](#). Then a test is made to see if there was a change in the environment. In the proposed algorithm, we use the best particle of the swarm for monitoring the environment in order to detect changes. At each iteration, before we update the swarm, the fitness of the best particle of the swarm is re-evaluated. If an alteration in the fitness of the best particle is observed, it indicates that the environment has changed during the last iteration. Once a change is detected in the environment, all particles are re-evaluated and t is set to zero. Afterwards, particles of the swarm update their velocity and position according to [Equation \(3\)](#) and [\(2\)](#), respectively.

4. Experimental setup

4.1 Dynamic test function

One of the most widely used synthetic dynamic optimisation test suites in the literature is the moving peaks benchmark (MPB) problem proposed by Branke ([2002](#)), which is highly regarded due to its configurability. MPB is a real-valued dynamic environment with a D -dimensional landscape consisting of N peaks, where the height, the width and the position of each peak is changed slightly every the time a change occurs in the environment ([Branke, 2002](#)). Different landscapes can be defined by specifying the shape of the peaks. A common conical peak is defined as follows:

$$f(\vec{x}, t) = \max_{i=1, \dots, N} H_t(i) - w_t(i) \sqrt{\sum_{j=1}^D (x_t(j) - x_t(i,j))^2}, \quad (5)$$

where $H_t(i)$ and $W_t(i)$ are the height and the width of peak i at time t , respectively. The coordinates of each dimension $j \in [1, D]$ related to the location of peak i at time t , are expressed by $X_t(i, j)$ and D is the problem dimensionality.

Every time a change occurs in the environment the height, the width and the position of each peak deviate from their current values according to the following equations:

$$H_{t+1}(i) = H_t(i) + \text{height}_{\text{severity}} \cdot \sigma, \quad (6)$$

$$W_{t+1}(i) = W_t(i) + \text{width}_{\text{severity}} \cdot \sigma, \quad (7)$$

$$\vec{X}_{t+1}(i) = \vec{X}_t(i) + \vec{v}_{t+1}(i), \quad (8)$$

$$\vec{v}_{t+1}(i) = \frac{s}{|\vec{r} + \vec{v}(i)|} ((1 - \lambda)\vec{r} + \lambda\vec{v}(i)), \quad (9)$$

where the shift vector $\vec{v}_{t+1}(i)$ is a linear combination of a random vector \vec{r} , which is created by drawing random numbers in $[-0.5, 0.5]$ for each dimension, and the current shift vector $\vec{v}_t(i)$, and normalised to the length s . Parameter $\lambda \in [0.0, 1.0]$ specifies the correlation of each peak's changes to the previous one. This parameter determines the trajectory of changes, where $\lambda = 0$ means that the peaks are shifted in completely random directions and $\lambda = 1$ means that the peaks always follow the same direction, until they hit the boundaries where they bounce off.

The environmental parameters used in the experiments of this article are set according to the values listed in [Table 1](#).

In order to test different environmental dynamics, we set the speed of changes $\sigma \in \{100, 200, 500\}$ and the severity of changes $s \in \{1.0, 2.0, 3.0, 4.0, 5.0\}$. This gives 15 different dynamic scenarios (i.e. 15 pairs of (σ, s)) which we use to validate the efficiency of our proposed method. [Table 2](#) shows different scenarios modelled by MPB on which the performance of PSO with three different inertia weight strategies will be tested.

4.2 Performance measure

The ability of PSO with different inertia weight strategies in tracking extrema on a specific scenario is measured by the distance between the best particle of the swarm and the highest peak in the search space, right before the change occurs in the environment ([Li & Dam, 2003](#)). The

[Table 1](#). Parameter settings for the MPB.

Parameter	Setting	Other tested values
Number of peaks (m)	1	5, 10, 50, 100, 200
Height severity	7.0	
Width severity	1.0	
Peak shape	Cone	
Number of dimensions (D)	5	10, 50, 100
Height range (H)	$\in [30.0 \ 70.0]$	
Width range (W)	$\in [1.0 \ 12.0]$	
Standard height (I)	50.0	
Search space range (A)	$[0.0 \ 100.0]^D$	
Frequency of change (σ)	500	100, 200
Shift severity (s)	1.0	2.0, 3.0, 4.0, 5.0
Correlation coefficient (λ)	$[0.0, 1.0]$	

Table 2. Different dynamic scenarios generated by MPB.

$\sigma \downarrow$	Scenarios				
100	1	2	3	4	5
200	6	7	8	9	10
500	11	12	13	14	15
s→	1.0	2.0	3.0	4.0	5.0

mean error of an algorithm is calculated as follows (Alizadeh, Meybodi, & Rezvanian, 2013; Kordestani, Rezvanian, & Meybodi, 2014; Ranginkaman, Kazemi Kordestani, Rezvanian, & Meybodi, 2014):

$$E_{\text{mean}} = \frac{1}{G} \times \sum_{i=1}^G \left(\frac{1}{N} \times \sum_{j=1}^N e_{ij} \right), \quad (10)$$

where $G = 100$ is the total number of environmental changes, $N = 30$ is the number of runs and e_{ij} is the distance between the best found position by the swarm and the highest peak in the last iteration at i th change and j th run. The smaller values of E_{mean} indicate the better ability of the optimization algorithm in tracking the optimum.

4.3 Experimental setting

Oscillating triangular inertia weight (Osc Tri), constant inertia weight (Const) and linearly decreasing inertia weight (Lin Dec) are applied to PSO with *gbest* model. For all three PSO variants, the number of particles in the swarm is set to 10 and the acceleration coefficients (learning factors) $c_1 = c_2 = 1.4962$ as suggested in (Clerc & Kennedy, 2002). Moreover, the velocity of particles is restrained by $v_{\max} = 5.0$. For constant inertia weight $w = 0.7298$ and for the other two strategies the range of $[w_{\min}, w_{\max}]$ is $[0.4, 0.9]$. Regarding linearly decreasing strategy, for each environment, the value of the inertia weight is decreased from w_{\max} to w_{\min} . For oscillating triangular inertia weight, the value of α is empirically selected from $[0.0, 0.5]$ and kept constant over all iterations. For each experiment, the results are averaged over 30 independent runs.

5. Experimental results

5.1 Effect of varying inertia weight

The experiment examines the effect of inertia weight on the performance of PSO in tracking the extrema on various dynamic scenarios. In this set of experiments, a peak is randomly initialised within the search domain and the population of particles is responsible for locating and tracking the peak changes in the landscape. The experimental results of PSO with different inertia weight strategies for several environmental conditions are reported in Tables 3–5.

As presented in the Tables 3–5, the performance of triangular inertia weight strategy is superior to other alternative methods. Figure 3 visualises the experimental results for different scenarios.

It is obvious from Figure 3 that PSO with triangular inertia weight outperforms the other approaches in all fifteen scenarios. From the results, it can also be seen that the performance of oscillating triangular inertia weight is less affected by the severity of changes. The reason lies in that the proposed inertia weight produces the waves of exploration and exploitation during the

Table 3. Performance of PSO with three different inertia weight strategy on MPB for $\sigma = 100$.

<i>s</i>	Const	Lin Dec	Osc Tri
1.0	3.1512	2.7068	2.1893
2.0	5.1164	4.7131	3.1480
3.0	4.8117	4.3060	3.5191
4.0	5.1835	4.7005	4.0113
5.0	5.8803	6.1615	4.6390

Note: The best result is highlighted in boldface.

Table 4. Performance of PSO with three different inertia weight strategy on MPB for $\sigma = 200$.

<i>s</i>	Const	Lin Dec	Osc Tri
1.0	2.3928	1.8238	1.5545
2.0	3.6754	2.8093	2.2751
3.0	3.2884	3.2832	2.1582
4.0	3.7624	3.6715	2.6842
5.0	3.9187	3.6200	2.8474

Note: The best result is highlighted in boldface.

Table 5. Performance of PSO with three different inertia weight strategy on MPB for $\sigma = 500$.

<i>s</i>	Const	Lin Dec	Osc Tri
1.0	2.2604	1.6881	1.3880
2.0	2.4667	2.2755	1.5990
3.0	2.6043	2.0823	1.4640
4.0	2.3298	1.9433	1.7391
5.0	2.5480	2.6160	1.8037

Note: The best result is highlighted in boldface.

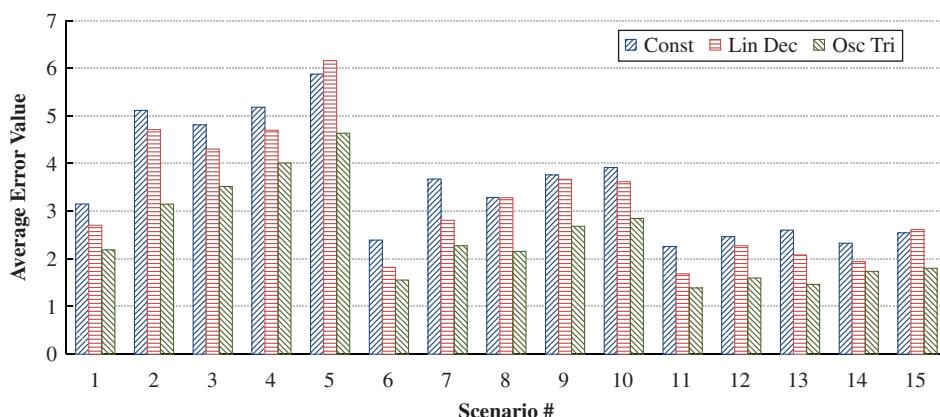


Figure 3. Average error value of three different inertia weight adjustment methods for 30 independent runs on 15 dynamic scenarios generated by MPB.

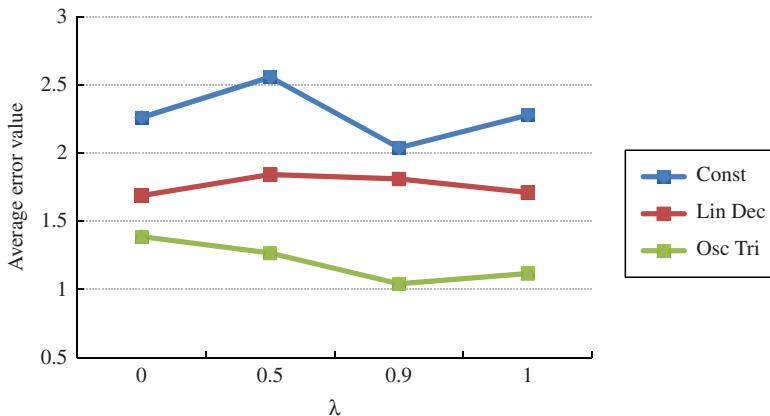


Figure 4. The effect of correlation shift on performance of PSO with different inertia weight strategies.

entire search process, which helps PSO to capture the new position of the peak in a less amount of time.

5.2 Effect of varying correlation of shifts

The effect of varying correlation of shifts on the performance of the PSO with different inertia weight methods is depicted in Figure 4. It can be understood from the Figure 4 that the performance of PSO is sensitive to the correlation of shifts. From Figure 4, it can be seen that oscillating triangular inertia weight outperforms the other two alternative strategies. Although PSO with linearly decreasing inertia weight is less affected by the correlation of shifts, it is beaten by the proposed oscillating triangular inertia weight significantly.

5.3 Effect of varying dimensions

In this set of experiments, we examine the influence of the higher dimension on the performance of different inertia weight strategies. It is obvious that increasing the number of dimensions of the environment causes the search space to be more complex. Therefore, the average error of the optimisation is expected to be higher as well. The results for different methods in dynamic environments with varying number of dimensions are given in Table 6. From Table 6, it can be seen that oscillating triangular inertia weight performs much better than the other two alternative methods in all test environments. In addition, it is worth noticing that the effect of the increasing dimensions upon oscillating triangular inertia weight is much lower than the influence upon other strategies.

Table 6. Performance of PSO with three different inertia weights for different number of dimensions.

D	Const	Lin Dec	Osc Tri
5	2.26040	1.68810	1.38800
10	3.57180	3.29530	2.62160
50	33.6235	37.4703	19.7013
100	55.3552	81.6784	31.7218

Note: The best result is highlighted in boldface.

Table 7. Effect of varying number of peaks on PSO with different inertia weight algorithm.

<i>m</i>	Const	Lin Dec	Osc Tri
1	2.26040	1.68810	1.38800
5	69.2526	70.0039	65.3607
10	77.1313	76.5533	74.8300
50	83.8064	80.0190	77.9184
100	79.6744	79.1321	74.8903
200	82.7668	78.0997	77.7706

Note: The best result is highlighted in boldface.

Table 8. Comparison of inertia weight algorithm and other adaptive PSO for varying number of peaks.

<i>m</i>	APSO	mQSO	Cellular PSO	Osc Tri
1	4.81	33.67	13.4	11.42
5	4.95	11.91	9.63	25.74
10	5.16	9.62	9.42	30.48
50	5.95	8.72	8.62	37.79
100	6.08	8.54	8.54	55.96
200	6.20	8.19	8.28	57.27

Note: The best result is highlighted in boldface.

5.4 Effect of varying number of peaks

In the last set of experiments, we compare the performance of oscillating triangular inertia weight with other two peer methods on the MPB with different number of peaks. Although we do not expect that simple PSO with inertia weight strategy produces promising results on multi-modal dynamic environments, we want to investigate the performance of our proposed method as the modality of the problem landscape increases. Table 7 shows the performance of PSO with three different inertia weight strategies in various dynamic environments where the number of peaks vary. As can be seen in Table 7, the performance of all three methods degrades significantly when the environment is multi-modal. However, the results achieved by the proposed method are slightly better than those achieved by the other strategies.

5.5 Comparison with other algorithms

In the last experiment, the offline error (Alizadeh et al., 2013; Kordestani et al., 2014; Ranginkaman et al., 2014) of the proposed method is compared with several well-known PSO variants including adaptive PSO as APSO (Rezazadeh, Meybodi & Naebi, 2011), multi-quantum swarm optimisation as mQSO (Blackwell & Branke, 2006) and cellular PSO (Hashemi & Meybodi, 2009b) for dynamic environment in Table 8. From Table 8, it is clear that the oscillating triangular inertia weight has a best result for single peaks, while for multi modal peaks, the performances of other algorithm is better than the inertia weight algorithm.

6. Conclusion and future works

Inertia weight parameter has a great influence on the behaviour of PSO. This parameter can be used to establish a balance between the global and local search ability of PSO. In this paper a

time-varying inertia weight, named oscillating triangular inertia weight, is proposed and its performance in terms of extrema tracking is compared with constant inertia weight and linearly decreasing inertia weight on moving peaks benchmark. Experimental results indicate that the proposed inertia weight improves the performance of PSO in tracking the changing optimum. Some future works include the adaptive selection of values for parameter α . It is also possible to research on other forms of oscillating functions like sinusoidal, cosine or asymmetric triangular function. Furthermore, different inertia weight functions can be applied to different particles at the same time. Finally, several mechanisms can be employed in PSO with the proposed inertia weight, e.g. multi-population approach, to conquer the existing challenges of dynamic environments.

Acknowledgements

The authors would like to thank the anonymous reviewers for their constructive comments to improve the quality of the paper.

Disclosure statement

No potential conflict of interest was reported by the authors.

References

- Aghdam, K. M., Mirzaee, I., Pourmahmood, N., & Aghababa, M. P. (in press). Design of water distribution networks using accelerated momentum particle swarm optimisation technique. *Journal of Experimental and Theoretical Artificial Intelligence*. doi:[10.1080/0952813X.2013.863227](https://doi.org/10.1080/0952813X.2013.863227)
- Alizadeh, M., Meybodi, M. R., & Rezvanian, A. (2013). Solving moving peak problem using a fuzzy particle swarm optimization based memetic algorithm. *The CSI Journal on Computer Science and Engineering*, 11, 10–21.
- Blackwell, T., & Branke, J. (2006). Multiswarms, exclusion, and anti-convergence in dynamic environments. *IEEE Transactions on Evolutionary Computation*, 10, 459–472.
- Branke, J. (2002). *Evolutionary optimization in dynamic environments*. Dordrecht: Kluwer.
- Cheng, H., & Yang, S. (2010). Genetic algorithms with immigrants schemes for dynamic multicast problems in mobile ad hoc networks. *Engineering Applications of Artificial Intelligence*, 23, 806–819. doi:[10.1016/j.engappai.2010.01.021](https://doi.org/10.1016/j.engappai.2010.01.021)
- Clerc, M., & Kennedy, J. (2002). The particle swarm – Explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6, 58–73. doi:[10.1109/4235.985692](https://doi.org/10.1109/4235.985692)
- Conforth, M., & Meng, Y. (2010). Reinforcement learning using swarm intelligence-trained neural networks. *Journal of Experimental and Theoretical Artificial Intelligence*, 22, 197–218. doi:[10.1080/09528130903065497](https://doi.org/10.1080/09528130903065497)
- Feng, C. S., Cong, S., & Feng, X. Y. (2007). A new adaptive inertia weight strategy in particle swarm optimization. *IEEE Congress on Evolutionary Computation*, 5, 4186–4190.
- Ghosh, S., Das, S., Kundu, D., Suresh, K., & Abraham, A. (2012). Inter-particle communication and search-dynamics of lbest particle swarm optimizers: An analysis. *Information Sciences*, 182, 156–168. doi:[10.1016/j.ins.2010.10.015](https://doi.org/10.1016/j.ins.2010.10.015)
- Gogna, A., & Tayal, A. (2013). Metaheuristics: Review and application. *Journal of Experimental and Theoretical Artificial Intelligence*, 25, 503–526. doi:[10.1080/0952813X.2013.782347](https://doi.org/10.1080/0952813X.2013.782347)
- Hasanzadeh, M., Meybodi, M. R., & Ebadzadeh, M. M. (2013). Adaptive cooperative particle swarm optimizer. *Applied Intelligence*, 39, 397–420. doi:[10.1007/s10489-012-0420-6](https://doi.org/10.1007/s10489-012-0420-6)
- Hasanzadeh, M., Meybodi, M. R., & Ghidary, S. S. (2011). Improving learning automata based particle swarm: An optimization algorithm. In *2011 IEEE 12th International symposium on computational intelligence and informatics (CINTI)* (pp. 291–296). Budapest: IEEE.

- Hashemi, A. B., & Meybodi, M. R. (2009a). A multi-role cellular PSO for dynamic environments. In *Proceedings of 14th international CSI computer conference* (pp. 412–417). Tehran: IEEE.
- Hashemi, A. B., & Meybodi, M. R. (2009b). Cellular PSO: A PSO for dynamic environments. *Advances in Computation and Intelligence, Lecture Notes in Computer Science*, 5821, 422–433.
- Hashemi, A. B., & Meybodi, M. R. (2011). A note on the learning automata based algorithms for adaptive parameter selection in PSO. *Applied Soft Computing*, 11, 689–705. doi:[10.1016/j.asoc.2009.12.030](https://doi.org/10.1016/j.asoc.2009.12.030)
- Kamosi, M., Hashemi, A. B., & Meybodi, M. R. (2010a). A hibernating multiswarm optimization algorithm for dynamic environments. In *Proceedings of 2010 second world congress on nature and biologically inspired computing (NaBIC 2010)* (pp. 363–369). Fukuoka: IEEE.
- Kamosi, M., Hashemi, A. B., & Meybodi, R. (2010b). A new particle swarm optimization algorithm for dynamic environments. *Swarm, Evolutionary, and Memetic Computing*, 6466, 129–138.
- Kennedy, J., & Eberhart, R. C. (1995). Particle swarm optimization. In *Proceedings of IEEE international conference on neural networks* (Vol. 4, pp. 1942–1948). Perth, WA: IEEE.
- Kentzoglaniakis, K., & Poole, M. (2009). Particle swarm optimization with an oscillating inertia weight. In *Proceedings of annual conference on genetic and evolutionary computation* (pp. 1749–1750). New York: ACM.
- Kordestani, J. K., Rezvanian, A., & Meybodi, M. R. (2014). CDEPSO: A bi-population hybrid approach for dynamic optimization problems. *Applied Intelligence*, 40, 682–694. doi:[10.1007/s10489-013-0483-z](https://doi.org/10.1007/s10489-013-0483-z)
- Li, X., & Dam, K. H. (2003). Comparing particle swarms for tracking extrema in dynamic environments. In *Proceedings of the IEEE congress on evolutionary computation (CEC 2003)* (pp. 1772–1779). Canberra: IEEE.
- Liu, L., Yang, S., & Wang, D. (2010). Particle swarm optimization with composite particles in dynamic environments. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 40, 1634–1648. doi:[10.1109/TSMCB.2010.2043527](https://doi.org/10.1109/TSMCB.2010.2043527)
- Nabizadeh, S., Faez, K., Tavassoli, S., & Rezvanian, A. (2010). A novel method for multi-level image thresholding using particle swarm optimization algorithms. In *Proceedings of the 2010 2nd international conference on computer engineering and technology (ICCET)* (pp. v4-271–v4-275). Chengdu: IEEE.
- Nabizadeh, S., Rezvanian, A., & Meybodi, M. R. (2012). Tracking extrema in dynamic environments using multi-swarm cellular PSO with local search. *International Journal of Electronic and Informatics*, 1, 29–37.
- Nickabadi, A., Ebadzadeh, M. M., & Safabakhsh, R. (2011). A novel particle swarm optimization algorithm with adaptive inertia weight. *Applied Soft Computing*, 11, 3658–3670. doi:[10.1016/j.asoc.2011.01.037](https://doi.org/10.1016/j.asoc.2011.01.037)
- Nickabadi, A., Ebadzadeh, M. M., & Safabakhsh, R. (2012). A competitive clustering particle swarm optimizer for dynamic optimization problems. *Swarm Intelligence*, 6, 177–206. doi:[10.1007/s11721-012-0069-0](https://doi.org/10.1007/s11721-012-0069-0)
- Norouzzadeh, M. S., Ahmadzadeh, M. R., & Palhang, M. (2012). LADPSO: Using fuzzy logic to conduct PSO algorithm. *Applied Intelligence*, 37, 290–304. doi:[10.1007/s10489-011-0328-6](https://doi.org/10.1007/s10489-011-0328-6)
- Ranginkaman, A. E., Kazemi Kordestani, J., Rezvanian, A., & Meybodi, M. R. (2014). A note on the paper ‘A multi-population harmony search algorithm with external archive for dynamic optimization problems’ by Turky and Abdullah. *Information Sciences*, 288, 12–14. doi:[10.1016/j.ins.2014.07.049](https://doi.org/10.1016/j.ins.2014.07.049)
- Rezaee Jordhei, A., & Jasni, J. (2013). Parameter selection in particle swarm optimisation: A survey. *Journal of Experimental and Theoretical Artificial Intelligence*, 25, 527–542. doi:[10.1080/0952813X.2013.782348](https://doi.org/10.1080/0952813X.2013.782348)
- Rezvanian, A., & Meybodi, M. R. (2010a). An adaptive mutation operator for artificial immune network using learning automata in dynamic environments. In *Proceedings of second world congress on nature and biologically inspired computing (NaBIC 2010)* (pp. 479–483). Fukuoka: IEEE.

- Rezvanian, A., & Meybodi, M. R. (2010b). Tracking extrema in dynamic environments using a learning automata-based immune algorithm. *Grid and Distributed Computing, Control and Automation*, 12, 216–225.
- Rezazadeh, I., Meybodi, M. R., & Naebi, A. (2011). Adaptive particle swarm optimization algorithm for dynamic environments. In Y. Tan, Y. Shi, Y. Chai, & G. Wang (Eds.), *Advances in swarm intelligence* (pp. 120–129). Berlin/Heidelberg: Springer.
- Shi, Y., & Eberhart, R. (1998). A modified particle swarm optimizer. In *Proceedings of the 1998 IEEE international conference on evolutionary computation (CEC 1998)* (pp. 69–73). Anchorage, AK: IEEE.
- Soleimani-Pour, M., Rezvanian, A., & Meybodi, M. R. (2012). Finding a maximum clique using ant colony optimization and particle swarm optimization in social networks. In *Proceedings of the international conference on advances in social networks analysis and mining (ASONAM 2012)* (pp. 58–61). Istanbul: IEEE Computer Society.
- Soleimani-Pour, M., Rezvanian, A., & Meybodi, M. R. (2014). An ant based particle swarm optimization algorithm for maximum clique problem in social networks. In F. Can, T. Özyer, & F. Polat (Eds.), *State of the art applications of social network analysis* (pp. 295–304). Switzerland: Springer International Publishing.
- van den Bergh, F. (2002). An analysis of particle swarm optimizers (Ph.D. dissertation). Department of Computer Science, University of Pretoria, Pretoria, South Africa.
- Wei, Z., Hou, J. Y., Tan, H., & Guo, G. N. (2010). Research on dynamic intrusion detection model based on risk coefficient. *Advanced Materials Research*, 129–131, 124–127. doi:[10.4028/www.scientific.net/AMR.129-131.124](https://doi.org/10.4028/www.scientific.net/AMR.129-131.124)
- Yang, S., & Li, C. (2010). A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments. *IEEE Transactions on Evolutionary Computation*, 14, 959–974. doi:[10.1109/TEVC.2010.2046667](https://doi.org/10.1109/TEVC.2010.2046667)
- Yazdani, D., Nasiri, B., Sepas-Moghaddam, A., & Meybodi, R. (2013). A novel multi-swarm algorithm for optimization in dynamic environments based on particle swarm optimization. *Applied Soft Computing*, 13, 2144–2158. doi:[10.1016/j.asoc.2012.12.020](https://doi.org/10.1016/j.asoc.2012.12.020)
- Zandieh, M., & Adibi, M. A. (2010). Dynamic job shop scheduling using variable neighbourhood search. *International Journal of Production Research*, 48, 2449–2458. doi:[10.1080/00207540802662896](https://doi.org/10.1080/00207540802662896)