Particle swarm optimization algorithm using exponential function -way-decreased inertia weight

Rim Ung Jang, Yong Chon Jang *, Se Yong Chon, Hak Mun Kim and Song Hak Hong

*Faculty of Information Science, **Kim Il Sung** University, Taesong District, Pyongyang, DPR Korea*

Abstract.

In this article we assumed that during the particle swarm optimization (PSO)process, the inertia weight value of the velocity vector calculating equation would be changed by non-liner way. And also this way reflects PSO's real nature very well. The inertia weight factor's non-liner-changed equation that is proposed is the flowing:

$$w = \left(\frac{1}{base}\right)^t \quad , \quad base \geq 2 \tag{1}$$

This equation is an exponential function.

Keywords: Particle swarm optimization (PSO), Convergence, Inertia weight factor

1. Introduction

PSO has been widely applied in many real optimization problems and has shown good performances, such as manufacturing control in engineering optimization [2,6-9], and multi-source scheduling in cloud computing [3-5].

Kennedy et al. firstly proposed PSO(Particle swarm optimization) which is a nature-inspired evolutionary algorithm[1].

It mimics the social behavior of bird flock and fish schools when they search for food [11].

PSO is an optimization method that utilizes swarm intelligences in solving problems.

It is very different from evolutionary algorithms in which various operators are applied to the population yielding next generation with higher fitness value (better solution).

In other words, the population evolves itself until it converges to the optimal solution in evolutionary algorithm.

In PSO, each particle is searching for the optimal solution therefore they are moving with a certain velocity.

Each particle also remembers the best result achieved so far (personal best) and exchanges information with other particles to determine the best particle (global best) among the swarm.

At each step, a particle has to move to a new position after adjusting its velocity.

* Corresponding author

Email addresses: *YC.JANG1122@star-co.net.kp* (YONG CHOL JANG)

The particle would tend to move towards its historical best position and the best position recorded by the swarm.

Hence, the velocity is actually comprised of the following three components:

(i) the current velocity,
(ii) weighted random portion in the direction of its personal best and
(iii) weighted random portion in the direction of global best.

The new position is merely the sum of current position with the new velocity.

In short, let the solution space be D-dimensional, then the i th particle in the swarm is

$$X_i = (X_{i1}, X_{i2}, ..., X_{iD})^T \tag{2}$$

The velocity vector is

$$V_i = (V_{i1}, V_{i2}, ..., V_{iD})^T \tag{3}$$

The historical best position of the i th particle is

$$P_i = (P_{i1}, P_{i2}, ..., P_{iD})^T \tag{4}$$

The best particle is recognized by fixing g as the index of the above expressions.

Standard PSO is only governed by the following two equations:

$$V_{id}{}^{t+1} = V_{id}{}^{t} + c_1 r_1{}^t\left(P_{id}{}^t - X_{id}{}^t\right) + c_2 r_2{}^t\left(P_{gd}{}^t - X_{id}{}^t\right) \tag{5}$$

$$X_{id}{}^{t+1} = X_{id}{}^{t} + V_{id}{}^{t+1} \tag{6}$$

In which (5) being velocity update equation and (6) being position update equation.

$$d=1,2,\ \dots\ , D; \qquad i=1,2,\ \dots\ , S \tag{7}$$

In the Eqs.(7), D is the dimension number and S is the swarm size. In the Eqs.(5), c1 and c2 are weight of personal best and weight of global best, respectively; r1 and r2 are random numbers distributed uniformly in [0,1].

The basic PSO has some drawbacks which make it trapped in the local optimum and suffer from the premature convergence.

How to improve the convergence speed as well as to avert the premature convergence has become the most important research problem in PSO.

So, Shi and Eberhart [12] proposed the inertia weight method based on the standard PSO algorithm in 1998.

The particle's velocity calculating equation in which the inertia weight item has been reflected is as follows.

$$V_{id}^{t+1} = wV_{id}^{t} + c_1 r_1^{t}\left(P_{id}^{t} - X_{id}^{t}\right) + c_2 r_2^{t}\left(P_{gd}^{t} - X_{id}^{t}\right) \quad (8)$$

Here, the inertia weight value is given from the following equation.

$$w = (w_{max} - w_{min}) * \frac{(t_{max} - t)}{t_{max}} + w_{min} \quad (9)$$

Initially, the higher setting value of w in the PSO algorithm will enhance the exploration of the particles into a wide scope, and as the value of w gets lower, due to the linear decrement, it will cause a higher exploitation of the particles to the local area.

According to Eberhart and Shi [10], the optimal strategy is to set w to 0.9 initially and then reduce it linearly to 0.4, allowing early exploration before exploiting in the proximity of global optimum later.

But, with the PSO's real nature, the inertia weight(w) should be decreased like non-linear way.

Fan and Chiu [13] proposed a nonlinearly decreasing weight method in which the equation to update the inertia weight is $w = (2/t)^{0.3}$, and in that equation t is the iteration number; w(t) is the inertia weight of t^{th} iteration.

But here, the inertia weight is changed between 1.23 to 0.4.

The inertia weight updating way should be more improved to reflect the real nature of PSO which imitate such as birds flying, fishes moving, or bees foraging.

Based on the results of observations on the swarm behaviors in some ecological system, from the initial time of swarm moving to the ending time of swarm moving, the possibility of depending on particle-self-exiting position should decrease like the exponential function.

So In this paper, in order to prove the above assumption, the way of using a non-linear decreasing inertia weight (w) which use the exponential function-way-decreased inertia weight is proposed.

The rest of this paper is organized as follows.

The experimental results and analysis which are got when PSO-0(PSO which is an initial PSO),PSO -1(PSO which uses the linear- decreased inertia weight) and PSO-2, PSO-3(PSO which uses the non-linear -decreased inertia weight) are applied on several benchmark multimodal functions are shown in Section 2,

The experimental results and analysis which are got when PSO-2 with several exponential bases are applied on several benchmark multimodal functions are shown in Section 3.

And the conclusion is provided in Section 4.

2. PSO which uses the exponential function -way-decreased inertia weight(w)

In case that the base of exponential function is a proper fraction, when the superscript is 0 the exponential function value is equal to 1, or when the superscript is increased than 0, the exponential function value is nearing to 0. For example, in case that the base is $\frac{1}{2}$, then the exponential function $w = (\frac{1}{2})^t$'s graph is as follows;

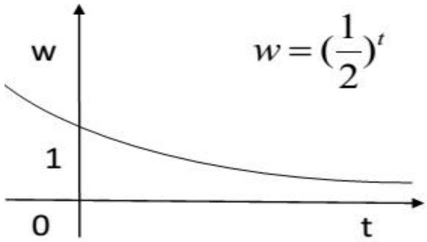


Fig.1. The graph of $w = (\frac{1}{2})^t$

This tendency of exponential functions 's graph reflects the tendency of a particle's velocity change through the processes of exploration and exploitation. So, it is true that the inertia weight's non-liner change should be made by the exponential function. Then, let's apply this to several benchmark functions and compare the result of this with one of PSO-1. In order to verify the performance of the algorithm, here we use 6 test functions which are shown in Table 1 .

Table 1. Test functions

number	Test functions	Range	Sorts
1	$f_1(x) = \sum_{i=1}^{n} x_i^2$	[-100, 100]	Unimodal functions
2	$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	[-10, 10]	Unimodal functions
3	$f_3(x) = \sum_{i=1}^{n} (\sum_{j=1}^{n} x_j)^2$	[-100, 100]	Unimodal functions
4	$f_4(x) = \sum_{i=1}^{n} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	[-10, 10]	Unimodal functions
5	$f_5(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	[-5.12, 5.12]	Multimodal functions
6	$f_6(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$	[-600, 600]	Multimodal functions

The first four functions (f1–f4) are unimodal functions and the next two functions are multimodal functions.

By doing experiments on these functions, we can verify that the proposed PSO-3 can maintain the fast convergence feature and have the ability of dealing with multimodal functions.

The conditions of experiments on these functions are shown in Table 2.

Table 2. Conditions of experiments on these functions

No	PSOs	Parameters
1	PSO-0	c_1=0.5 +ln2, c_2=0.5 + ln2 w =1 / (2 * ln2) r_1 , r_2 : random numbers uniformly distributed within [0,1]
2	PSO-1	c_1=0.5 +ln2, c_2=0.5 + ln2 r_1 , r_2 : random numbers uniformly distributed within [0,1] Inertia weight decreasing way: linearly decrease from w_{max}=0.9 to w_{min}=0.4
3	PSO-2	c_1=0.5 +ln2, c_2=0.5 + ln2 r_1 , r_2 : random numbers uniformly distributed within [0,1] Inertia weight decreasing way: *non*-linearly decrease from w_{max}=1.2 to w_{min}=0.4 as $w = (2/t)^{0.3}$
4	PSO-3	c_1=0.5 +ln2, c_2=0.5 + ln2, *base*=2 r_1 , r_2 : random numbers uniformly distributed within [0,1] Inertia weight decreasing way : *non*-linearly decrease from w_{max}=1 to w_{min}=0 as an exponential function

The results of comparison on 6 basic functions are shown in Table 3.

To make comparing experimental results of using different algorithms clear, each algorithm will run on the corresponding benchmark function independently CN=100 times and the mean error of results will be displayed in the tables of comparison results below.

The population size *n* in each algorithm is set as 20 and the repeated number T is 1000.

The comparison results on convergence accuracy including the mean error ($MEAN_o$) , the error's standard deviation(σ_o) and the average of swarm repeating number t at which the optimization value first is searched(FTA_o) are listed in Tables 3.

Table 3　Results comparison on 6 basic functions

<table>
<tr><th colspan="3">PSOs</th><th>PSO-0</th><th>PSO-1</th><th>PSO-2</th><th>PSO-3</th></tr>
<tr><td rowspan="12">단일극값
함수들</td><td rowspan="3">f_1</td><td>$MEAN_o$</td><td>1.0137E-99</td><td>3.0837E-150</td><td>0</td><td>**0**</td></tr>
<tr><td>σ_o</td><td>3.5817E-99</td><td>7.5275E-150</td><td>0</td><td>**0**</td></tr>
<tr><td>FTA_o</td><td>996</td><td>982</td><td>620</td><td>**273**</td></tr>
<tr><td rowspan="3">f_2</td><td>$MEAN_o$</td><td>9.19871E-48</td><td>9.00088E-75</td><td>9.035E-286</td><td>**0**</td></tr>
<tr><td>σ_o</td><td>3.11833E-47</td><td>2.45964E-74</td><td>0</td><td>**0**</td></tr>
<tr><td>FTA_o</td><td>996</td><td>974</td><td>1000</td><td>**562**</td></tr>
<tr><td rowspan="3">f_3</td><td>$MEAN_o$</td><td>1.01283E-93</td><td>8.1113E-147</td><td>0</td><td>**0**</td></tr>
<tr><td>σ_o</td><td>3.65099E-93</td><td>2.9214E-146</td><td>0</td><td>**0**</td></tr>
<tr><td>FTA_o</td><td>992</td><td>982</td><td>626</td><td>**287**</td></tr>
<tr><td rowspan="3">f_4</td><td>$MEAN_o$</td><td>0</td><td>0</td><td>0</td><td>**0**</td></tr>
<tr><td>σ_o</td><td>0</td><td>0</td><td>0</td><td>**0**</td></tr>
<tr><td>FTA_o</td><td>332</td><td>111</td><td>104</td><td>**33**</td></tr>
<tr><td rowspan="6">다극값
함수들</td><td rowspan="3">f_5</td><td>$MEAN_o$</td><td>0</td><td>0</td><td>0</td><td>**0**</td></tr>
<tr><td>σ_o</td><td>0</td><td>0</td><td>0</td><td>**0**</td></tr>
<tr><td>FTA_o</td><td>185</td><td>61</td><td>67</td><td>**22**</td></tr>
<tr><td rowspan="3">f_6</td><td>$MEAN_o$</td><td>0</td><td>0</td><td>0</td><td>**0**</td></tr>
<tr><td>σ_o</td><td>0</td><td>0</td><td>0</td><td>**0**</td></tr>
<tr><td>FTA_o</td><td>211</td><td>69</td><td>74</td><td>**22**</td></tr>
</table>

$MEAN_o$, σ_o, FTA_o are obtained by the following Eqs. 10 -12.

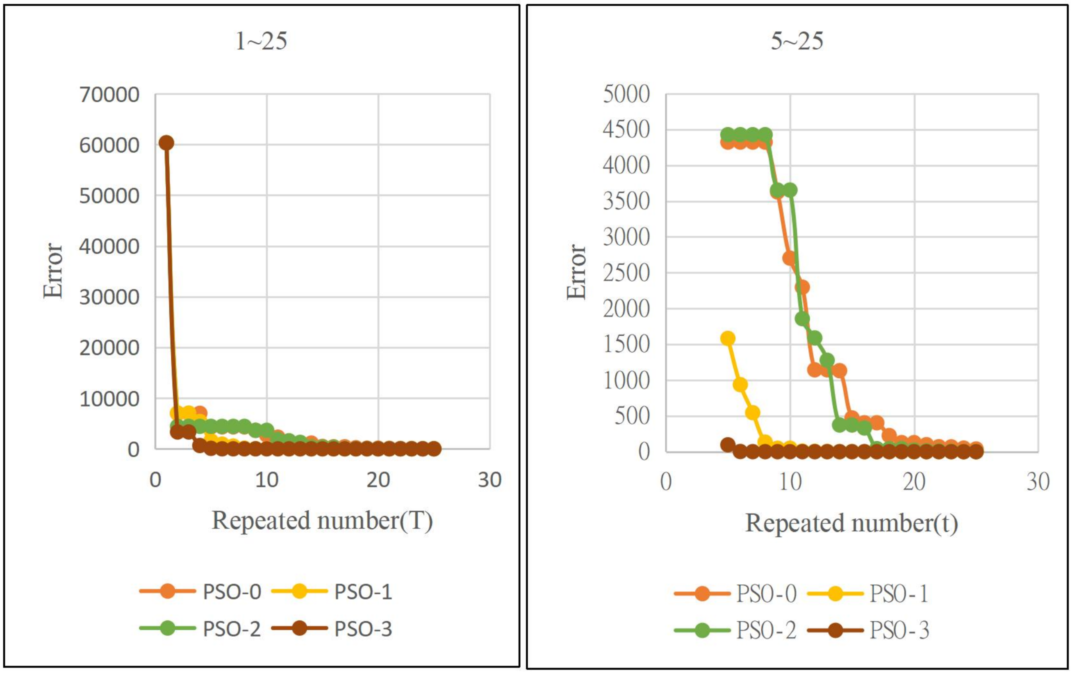
$$MEAN_O = \frac{1}{CA}\sum_{i=1}^{CA} Error_i \qquad (10)$$

$$\sigma_o = \sqrt{\frac{1}{CA}\sum_{i=1}^{CA}(Error_i - MEAN_o)^2} \qquad (11)$$

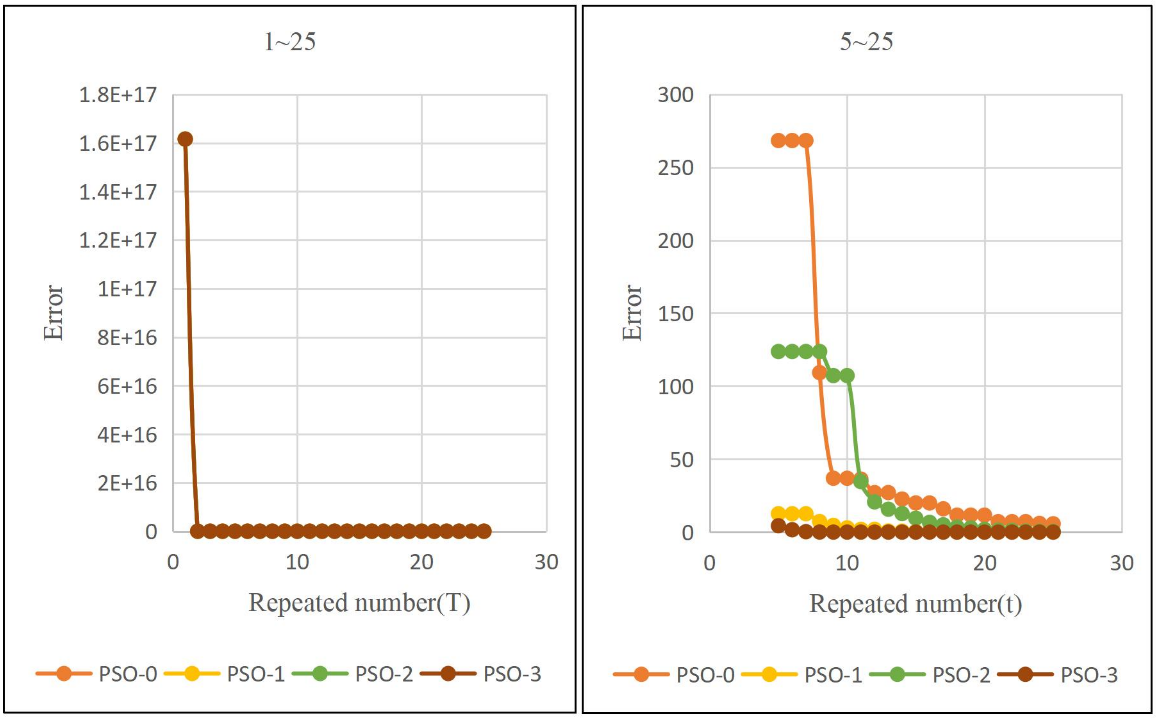
$$FTA_o = \frac{1}{CA}\sum_{i=1}^{CA} FT_i \qquad (12)$$

CA : the number of running
$Error_i$: Error at the i^{th} running
FT_i : Swarm repeating number t at which the optimization value first is searched at the i^{th} running

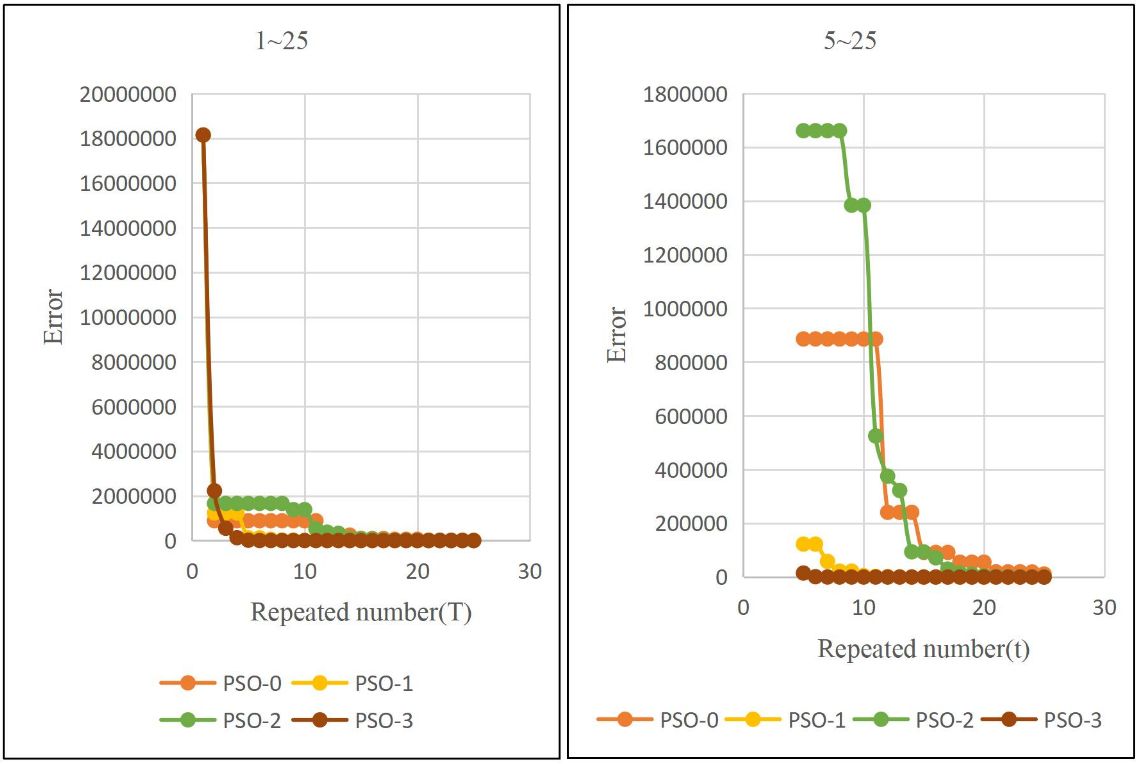
Here, we mark the best results performed by those algorithms on each test functions with bold font, and mark the second best results of them with underlined. From values with bold font in Tables 3 , we can obviously see that PSO-3 performs well on all benchmark functions, unimodal and multimodal functions (f1–f6) which are used in this experiment.



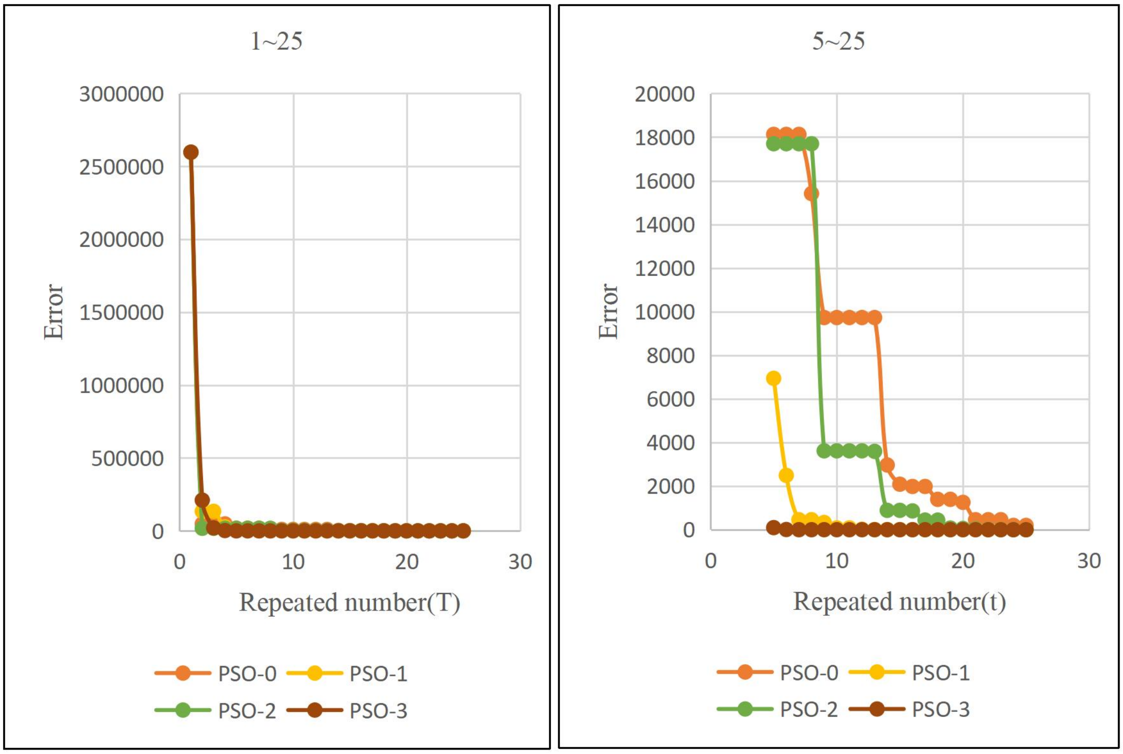
a) Convergence process on f_1



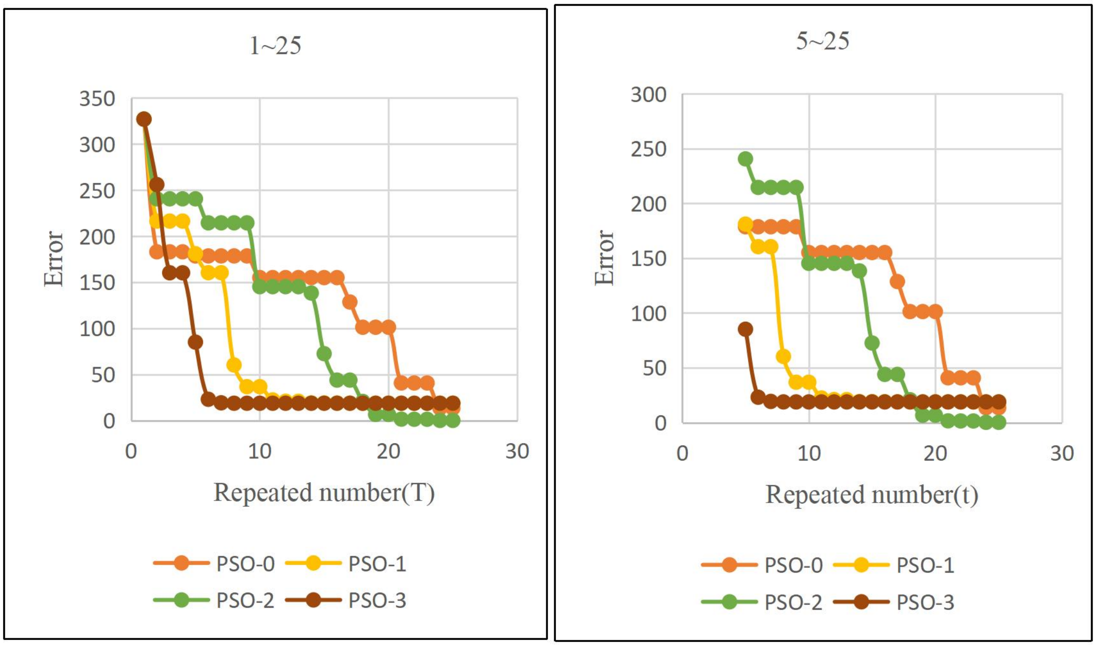
b) Convergence process on f_2



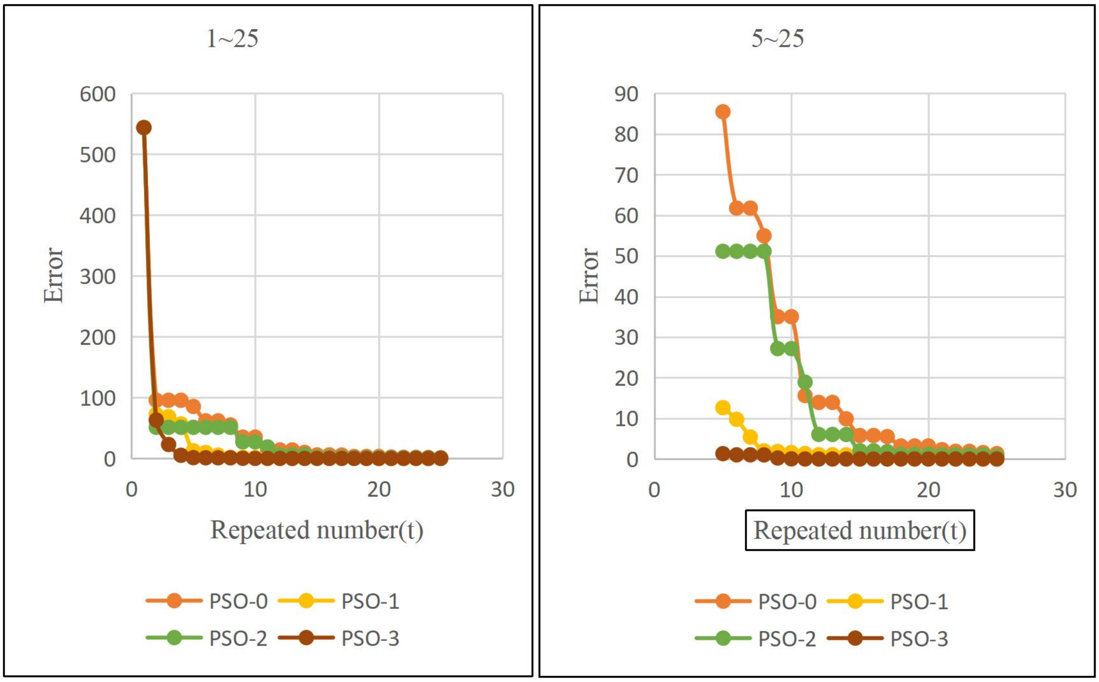
c) Convergence process on f_3



d) Convergence process on f_4



e) Convergence process on f_5



f) Convergence process on f_6

Fig. 2. Convergence process on 6 basic multimodal functions .

Fig. 2 shows the convergence process of the algorithms. From the above figures, we can see that the proposed PSO-3 can converge with an ideal convergence speed, especially on multimodal functions and has the fastest convergence speed among these algorithms.

3. Effects of exponential function's base-change

The experimental results and analysis which are got when PSO-3 with several exponential bases($\frac{1}{2}$, $\frac{1}{5}$, $\frac{1}{10}$, $\frac{1}{13}$, $\frac{1}{15}$) are applied on several benchmark multimodal functions are shown in Section 3.

The benchmark multimodal functions that are used in this experiment are equal to Table1. Conditions of experiment is equal to Table4. The result of this experiment is showed in Table5.

Table 4. Conditions of experiments for choosing the good base value

PSO	Parameters
PSO-3	c_1=0.5 +ln2, c_2=0.5 + ln2 r_1 , r_2 : random numbers uniformly distributed within [0,1] Inertia weight decreasing way : *non*-linearly decrease from w_{max}=1 to w_{min}=0 as an exponential function

Table 5. Each base value's optimization ability order analysis result - 1

Base	Index	Benchmark functions					
		f1	f2	f3	f4	f5	f6
1 - 2	$MEAN_o$	1.14E-18	9.32E-11	2.25E-84	0	15.75352	0
	σ_o	2.8E-18	2.28E-10	5.52E-84	0	7.71762	0
	$0.5*(MEAN_o+\sigma_o)$	1.97E-18	1.61E-10	3.89E-84	0	11.73557	0
	Order	4	3	3	1	4	1
1 - 7	$MEAN_o$	0.005819	8.68E-62	2.2E-187	0	12.60282	3.76E-05
	σ_o	0.014255	2.13E-61	0	0	9.762102	9.21E-05
	$0.5*(MEAN_o+\sigma_o)$	0.010037	1.5E-61	1.1E-187	0	11.18246	6.48E-05
	Order	5	1	1	1	3	2
1 - 10	$MEAN_o$	8.57E-26	4.58E-05	2.12E-32	0	6.30141	0
	σ_o	2.1E-25	0.000112	5.19E-32	0	9.762102	0
	$0.5*(MEAN_o+\sigma_o)$	1.48E-25	7.9E-05	3.66E-32	0	8.031756	0
	Order	2	4	4	1	1	1
1 - 13	$MEAN_o$	2.23E-42	5.12E-52	1.09E-28	0	9.452115	0
	σ_o	5.46E-42	1.25E-51	2.66E-28	0	10.35427	0
	$0.5*(MEAN_o+\sigma_o)$	3.84E-42	8.83E-52	1.88E-28	0	9.903194	0
	Order	1	2	5	1	2	1
1 - 15	$MEAN_o$	1.57E-19	0.180952	1.8E-101	0	6.30141	0
	σ_o	3.66E-19	0.443241	4.3E-101	0	9.762102	0
	$0.5*(MEAN_o+\sigma_o)$	2.61E-19	0.312097	3E-101	0	8.031756	0
	Order	3	5	2	1	1	1

In Table5, $0.5*(MEAN_o+\sigma_o)$ is used to reflect $MEAN_o$ andσ_o equally

Each base value order of Each Benchmark function's optimum value searching is estimated by this index. The result is Table6.

Table6. Each base value's optimization ability order analysis result - 2

Base	Benchmark functions						Average	σ_A	$0.5*(MEAN_o+\sigma_o)$	Order
	f1	f2	f3	f4	f5	f6				
1/2	4	3	3	1	4	1	2.666667	1.36626	2.016463	5
1/7	5	1	1	1	3	2	2.166667	1.602082	1.884374	4
1/10	2	4	4	1	1	1	2.166667	1.47196	1.819313	2
1/13	1	2	5	1	2	1	2	1.549193	1.774597	1
1/15	3	5	2	1	1	1	2.166667	1.602082	1.884374	3

From the above Table6, we can see that in the case of base $\frac{1}{13}$, the proposed PSO-3 can converge with an convergence speed .

So, The inertia weight factor's non-liner-changed equation that is proposed is the flowing:

$$w=(\frac{1}{13})^t \tag{13}$$

4. Conclusion and future work

In this paper, we present a Parameter modification based PSO algorithm, which focus on the modification or adjustment methods of the inertia weights to reflect more about the nature of PSO.

We mainly focus on inertia weight factor's non-liner-changed equation and particles' local search strategy learning. The inertia weight factor's non-liner-changed equation is an exponential function. In order to verify the performance of the algorithm which is using the inertia weight factor's non-liner-changed equation, here we use 6 benchmark functions.

The results of this algorithm comparing with 3 algorithms show that it can get better convergence accuracy as well as faster convergence speed in most cases.

The experimental results show the competitive performance of this algorithm and the ability to solve complex problems such as multimodal test problems and composition test problems is verified.

As the problems in reality are getting more and more complicated, in the future, we will further try to investigate the applications of this algorithm into solving various challenging optimization problems in reality.

ACKNOWLEDGMENTS

The author would like to thank all the anonymous reviewers for their valuable comments and criticism that significantly improved the quality of the article.

References

[1] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of the IEEE International, Conference on Neural Networks, 4(2002) 1942–1948.

[2] N. Kalaiarasi, S. Paramasivam, S.S. Dash, P. Sanjeevikumar, L. Mihet-Popa, PSO based MPPT implementation in d space controller integrated through z-source inverter for photovoltaic applications, Energies , 9(2017) 1–10.

[3] J. Li, Y. Zhang, X. Chen, Y. Xiang, Secure attribute-based data sharing for resource-limited users in cloud computing, Comput. Secur. 72 (2018) 1–12.

[4] P. Li, J. Li, Z. Huang, T. Li, C.-z. Gao, W.-B. Chen, K. Chen, Privacy-preserving outsourced classification in cloud computing, Cluster Comput. (2017) 1–10.

[5] P. Li, J. Li, Z. Huang, T. Li, C.-z. Gao, S.-M. Yiu, K. Chen, Multi-key privacy-preserving deep learning in cloud computing, Future Gener. Comput. Syst. 74(2017) 76–85.

[6] T.K. Maji, P. Acharjee, Multiple solutions of optimal PMU placement using exponential binary PSO algorithm for smart grid applications, IEEE Trans. Ind. Appl. 53 (3) (2017) 2550–2559.

[7] V.K. Pathak, A.K. Singh, A particle swarm optimization approach for minimizing GDT error in additive manufactured parts: PSO based GDT minimization, Int. J. Manuf. 7 (3) (2017) 69–80.

[8] F. Wang, Y. Zhang, Q. Rao, K. Li, H. Zhang, Exploring mutual information-based sentimental analysis with kernel-based extreme learning machine for stock prediction, Soft Comput. 21 (12) (2017) 3193–3205

[9] H. Wu, L. Kuang, F. Wang, R. Qi, G. Maoguo, L. Yuanxiang, A multiobjective box-covering algorithm for fractal modularity on complex networks, Appl. Soft. Comput. 61 (2017) 294–313.

[10] Eberhart RC, Shi y. Comparing inertia weights and constriction factors in particle swarm optimization. Congress on Evolutionary Computing 1(2000) 84–88.

[11] Eberhart RC, Kennedy J. A new optimizer using particle swarm theory. In: Proceedings of the sixth international symposium on micromachine and human science. Nagoya, Japan, (1995) 39–43.

[12] Shi Y, Eberhart RC. A modified particle swarm optimizer. In: Proceedings of IEEE world congress on computational intelligence (1998) 69–73.

[13] S.K.S. Fan, Y.Y. Chiu, A decreasing inertia weight particle swarm optimizer, Eng. Optim. 39 (2007) 203–228.