Exponential Inertia Weight for Particle Swarm Optimization

T.O. Ting[1,*], Yuhui Shi[1], Shi Cheng[2], and Sanghyuk Lee[1]

[1] Dept. Electrical and Electronic Engineering,
Xi'an Jiaotong-Liverpool University, Suzhou, China
[2] Dept. Electrical Engineering and Electronics,
University of Liverpool, Liverpool, UK
toting@xjtlu.edu.cn

Abstract. The exponential inertia weight is proposed in this work aiming to improve the search quality of Particle Swarm Optimization (PSO) algorithm. This idea is based on the adaptive crossover rate used in Differential Evolution (DE) algorithm. The same formula is adopted and applied to inertia weight, *w*. We further investigate the characteristics of the adaptive *w* graphically and careful analysis showed that there exists two important parameters in the equation for adaptive *w*; one acting as the local attractor and the other as the global attractor. The 23 benchmark problems are adopted as test bed in this study; consisting of both high and low dimensional problems. Simulation results showed that the proposed method achieved significant improvement compared to the linearly decreasing method technique that is used widely in literature.

Keywords: Benchmark functions, exponential inertia weight, Particle Swarm Optimization.

1 Introduction

Inertia weight, *w* has been one of the important parameters in Particle Swarm Optimization (PSO) algorithm. It has been known that *w* plays a crucial role in guaranteeing the robustness of PSO. Y. Shi and R. Eberhart first introduced the concept of *w* in PSO [1]. To date, a myriad of investigations concerning this parameter has been carried out [2-4]. The work by Bansal et al. [3] compared 15 inertia weight strategies available from literature. However, only 5 benchmark functions are employed in his work. Han et al. compares several inertia weights in his work [4]. Again, using only 3 benchmark problems is not adequate to validate the results obtained and conclusions made may not be true when more benchmark problems are considered. The improvement contributed by manipulation of *w* can be categorized into few categories, namely exploration and exploitation, mutating *w* and adaptive *w*.

* Corresponding author.

Y. Tan, Y. Shi, and Z. Ji (Eds.): ICSI 2012, Part I, LNCS 7331, pp. 83–90, 2012.

The first category, exploration and exploitation is based on the concept of incorporating high value of *w* and decreasing its value along the iteration. When *w* is high, the algorithm is capable of global search and as *w* decreases, the local search capability is more significant. This concept is implemented as linearly decreasing *w* as proposed by Shi in [1]. This technique is by far the most popular one and has been applied successfully in many works [5-7]. Many other variants are built upon this concept. Xin et al. introduced multi-stage linearly decreasing inertia weight[8]. In [9], instead of decreasing *w* from 0.9 to 0.4, the *w* is increased from 0.4 to 0.9. The range of variation of *w* is within 0.9 to 0.4 in his work.

The second category of improvement via *w* involves manipulation of *w* in a stochastic manner. Miranda proposed mutated *w* in [10] for reliability calculation in power systems. Feng proposed the chaotic *w* [11-12] based on the linearly decreasing *w* and random *w*. The stochastic mutation of *w* is introduced by Li in [13]. The method is performed along with the linearly decreasing *w*. There are in fact limited works under this category as the stochastic strategies introduce disturbances into the algorithm and these may not perform well on a wide range of problems.

Lastly, the third category implements *w* in an adaptive manner. This is perhaps the trend in many current works. Many works proposed ways to incorporate the information such as ranking [14], diversity [15], convergence, and swarm size[16] into *w* as this will dynamically adjust *w* based on the performance criteria received from the population. Work by Chen [16] relates the inertia weight with problem complexity (dimension size) and population size. If the swarm size is small, a larger inertia weight is employed to improve the global search capability to locate the global optimum. For an optimization problem on multi-dimension complex solution space, a larger inertia weight is employed to strengthen the ability to escape from local optima.

Many works on the inertia weight has been done, however, there is not clear justification of how this parameter can be adjusted to improve the performance of PSO algorithm. Thus, we aim to investigate this property in this work. The rest of the paper is organized as follows. Section 2 explains the proposed method. Parameter settings are described in Section 3. The benchmark problems used as the test bed are described under this section. This is followed by results and discussions in Section 4 and finally the conclusions in Section 5.

2 Proposed Exponential Inertia Weight, *w*

The idea of adaptive *w* in this paper originated from the work by Ao and Chi in [17]. In this reference, the author proposes an Adaptive Crossover Rate, *ACR* for Differential Evolution (DE) algorithm. This *ACR* is defined as:

$$CR = CR_0 \cdot e^{-a\left(\frac{g}{G}\right)^b} \tag{1}$$

where CR_0 is the initial crossover rate = 0.8 or 0.85, g is the current generation number, G is the maximum number of generations, a = 2, b = 2 or 3. The adaptive function for the crossover rate is simply crafted based on the logic of high CR at the early of run to prevent premature convergence and low CR at the end of run to enhance the local search. This concept is exactly the same for the case of inertia weight, w in PSO. Thus, the ACR is converted to adaptive w as follows:

$$w = w_0 \cdot e^{-a\left(\frac{g}{G}\right)^b} \tag{2}$$

whereby w_0 is set to 0.9 in our work here. This value is chosen as an initial value in many works [5-8]. Further, two graphs are plotted, depicting the characteristics of parameters a and b. These are plotted in Figs. 1 and 2 below. From Fig. 1, by increasing a from 0 to 3 with a step of 0.5, it has the ability to push down the value of w, resulting in a curvilinear curve along the iterations. Thus, we name the parameter a as local search attractor. Note that when a=0, the inertia weight becomes a static value of 0.9 as the initial value w_0 is set to 0.9. In the same diagram, take note that the third curve (a=1.0) starts from 0.9 and ends at approximately 0.32; almost similar to linearly decreasing w.

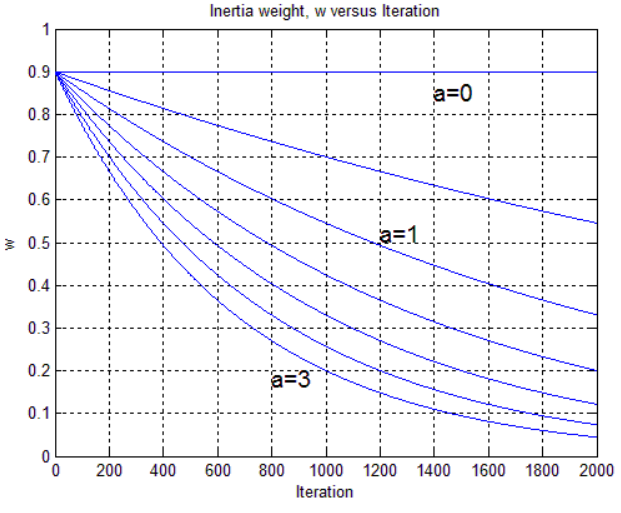

Fig. 1. Characteristics of local search attractor, a varies from 0 to 3 step 0.5 while b is set to 1

On the other hand, the parameter b has the opposite characteristic. When b is increased, it has the ability to pull up the curve resulting in higher value of w at the early run along the iterations. Hence, b is called global search attractor in this context. Again, note that when b=0, it is in fact a static w with the value approximate to 0.32. The third curve from below (b=1) is exactly the same as the curve (a=1) in Fig. 1 as both has the same numerical values (a=1, b=1).

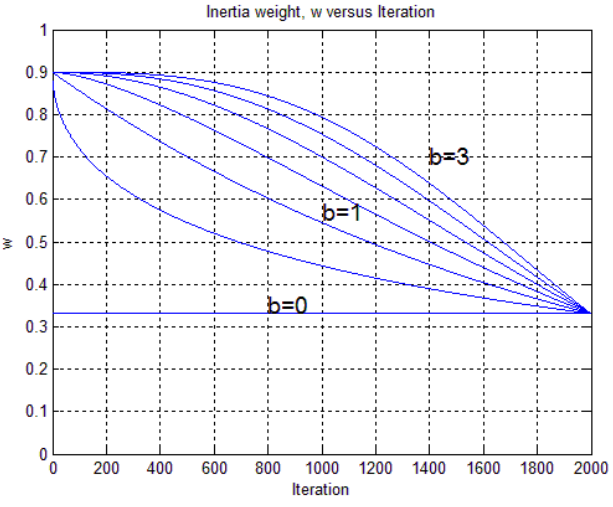

Fig. 2. Characteristics of global search attractor, b varies from 0 to 3 step 0.5 while a is set to 1

Generally, the ability of pulling up and pushing down the value of w using (2) makes the propose method ideal in PSO. Implementing this operator into any program is just a simple task and it does not add significant additional computational cost to the PSO algorithm.

3 Parameter Settings

The following settings are adopted in the PSO algorithm applied in this work. The population size is set to 20, acceleration coefficients; c_1 and c_2 are both set to 2.0. A dimensional value is reinitialized upon violation of either upper or lower boundaries. No maximum velocity, V_{max} is imposed in this setting. The results of linearly decreasing inertia weight are applied as standard comparison for the exponential w. The setting for exponential w is tabulated in Table 1.

Table 1. Setting for for exponential w

Method	Setting	Method	Setting
A	a=1, b=0.5	F	a=0.5, b=1
B	a=1, b=1.0	G	a=1.0, b=1
C	a=1, b=1.5	H	a=1.5, b=1
D	a=1, b=2.0	I	a=2.0, b=1
E	a=1, b=2.5	J	a=2.5, b=1

The widely known 23 benchmark problems [18] are adopted as test bed to validate the effectiveness of exponential w proposed in this work. All the parameter settings applied are similar. Results recorded as mean and standard deviation from 50 trials.

4 Results and Discussions

Results above are the summary of the performance of each method compared to the results of standard PSO. A shaded cell is visible when the mean obtained is better or equal to the standard PSO's. Otherwise, the cell is left empty instead of 0 for better readability. The total number of results that outperform the standard PSO is depicted in the last row of the table. The numerical values of the mean are available in Table 3.

Table 2. Results of simulation using different settings of a and b

	f	Function name	SPSO (mean)	δ	a is fixed at 1.0					b is fixed at 1.0				
					A	B	C	D	E	F	G	H	I	J
High Dimensional f	f_1	*Sphere*	6.61E-05	0.00	■	■	■				■	■	■	■
	f_2	*Schwefel 2.22*	4.19E-06	0.00	■	■	■				■	■	■	■
	f_3	*Schwefel 1.2*	35.98	25.84	■	■	■				■	■	■	■
	f_4	*Schwefel 2.21*	5.34	1.93	■	■	■	■			■	■	■	■
	f_5	*Rosenbrock*	26.69	31.27	■	■	■	■	■	■	■	■	■	■
	f_6	*Step*	8.00E-01	1.11	■	■					■		■	■
	f_7	*Quartic*	3.73E-02	0.01	■	■	■				■	■	■	■
	f_8	*Schwefel*	-7303.35	1132.26	■	■	■	■	■		■	■	■	■
	f_9	*Rastrigin*	24.36	6.42										
	f_{10}	*Ackley*	0.23	0.53										
	f_{11}	*Griewank*	1.55E-02	0.02		■							■	
	f_{12}	*Penalized P8*	4.00E-01	0.59	■	■	■				■	■	■	■
	f_{13}	*Penalized P16*	1.65E-01	0.52	■	■	■				■	■	■	■
Low Dimensional f	f_{14}	*Foxholes*	1.18	0.62										
	f_{15}	*Kowalik*	3.07E-04	0.00	■	■	■				■	■	■	■
	f_{16}	*Six-hump Camel-Back*	-1.0316280	0.00										
	f_{17}	*Branin*	4.02E-01	0.01	■	■	■				■	■	■	■
	f_{18}	*Goldstein-Price*	3.0000001	0.00	■	■	■				■	■	■	■
	f_{19}	*Hartman-3*	-3.8622	0.00	■	■	■	■			■	■	■	■
	f_{20}	*Hartman-6*	-3.2500	0.09	■	■					■	■	■	■
	f_{21}	*Shekel-5*	-5.89	3.48		■	■	■	■	■	■		■	
	f_{22}	*Shekel-7*	-6.78	3.57		■	■	■	■	■			■	
	f_{23}	*Shekel-10*	-7.80	3.57				■	■	■		■		
Total improvements, $\sum$					16	19	16	7	5	4	17	16	19	17

Results from the simulation above show that the use of proposed w is effective in tackling global optimum as generally majority of the methods outperform linearly decreasing method. This is true for methods A, B, C, G, H, I, J whereby 15 and above benchmark functions are solved with improvement. From left to right for methods A-E, as the global attractor, b is increased, the algorithm lack convergence capability. This is due to the reason that as b increases, the value of w increases and thus the algorithm is capable of global optimum and lack convergence capability. Note that f_{21}, f_{22} and f_{23} favor higher w to find the global optimum more accurately. Again, numerical values for these results are tabulated in Tables 3.

To ease our analysis, methods A-J are grouped into three categories, namely global, balance and local categories:

Global search ($b > a$) : Methods C, D, E and F
Balance search ($a = b$) : Methods B and H
Local search ($a > b$) : Methods A, H, I and J

The grouping of the methods above is based on the concept that when the global attractor is greater than the local attractor ($b>a$), the PSO algorithm is capable of global search. Similarly, the algorithm tend to be local searcher when the local attractor is greater than the global attractor ($a>b$). The balance group has the same value for both attractors. Results for each of these groups are recorded in Table 3. For each numerical value, the shading denotes the degree of the results obtained for each benchmark problem among all participating methods. Hereby, brighter background shading denotes better results.

At a glance of Table 3, it is easy to come to a conclusion that the local category methods are favored in this case as in this category majority of the mean recorded are above average (brighter shading). However the drawback of local category is the danger of being trapped in local optima. This is true for the case of f_{21}, f_{22} and f_{23}. For simplicity, we would propose the use of balance method (B and H). Note that B and H are both the same as a and b are both set to unity. In this category, there is a balance between global exploration and local exploitation.

Table 3. Results using global search methods (Methods C, D, E, F and G)

	Global Search Methods				Balance	Local Search Methods			
f	C	D	E	F	B / H	A	H	I	J
f_1	2.21E-06	1.44E-04	1.44E-03	4.63E-01	4.50E-10	2.09E-15	6.93E-14	1.82E-16	1.66E-14
f_2	5.35E-07	1.39E-04	1.10E-04	3.43E-03	1.34E-09	1.32E-10	4.39E-08	1.50E-06	9.14E-06
f_3	11.98	45.96	108.88	828.6	1.98	0.03	0.1	0.08	0.25
f_4	3.76	4.91	6.68	14.81	2.28	0.55	0.87	1.06	1.38
f_5	20.78	18.02	19.35	20.35	12.13	12.72	20.65	18.86	24.33
f_6	8.80E-01	2.40E+00	3.26E+00	3.68E+00	3.20E-01	4.80E-01	2.56E+00	5.60E-01	6.80E-01
f_7	2.97E-02	4.24E-02	4.90E-02	7.18E-02	2.31E-02	1.84E-02	2.14E-02	2.07E-02	2.35E-02
f_8	-8434.94	-8053.32	-7777.45	-4931.8	-8948.23	-9007.49	-8764.27	-8834.95	-8908.41
f_9	27.24	26.93	28.48	24.55	27.08	29.13	27.52	29.35	31.68
f_{10}	0.64	0.6	0.88	1.21	0.38	0.51	1.05	1.09	1.22
f_{11}	1.88E-02	1.80E-02	1.69E-02	7.88E-02	1.52E-02	1.73E-02	1.95E-02	1.51E-02	1.34E-02
f_{12}	1.88E-01	8.18E-01	1.24E+00	5.15E+00	2.34E-01	1.89E-01	2.12E-01	1.50E-01	2.66E-01
f_{13}	1.25E-01	1.87E-01	1.62E+00	1.19E+01	3.19E-02	7.67E-03	3.47E-02	6.67E-02	2.04E-01
f_{14}	1.36	1.22	1.41	1.24	1.3	1.43	1.38	1.29	1.1
f_{15}	3.07E-04	3.08E-04	3.08E-04	3.12E-04	3.07E-04	3.07E-04	3.07E-04	3.07E-04	3.07E-04
f_{16}	-1.03163	-1.03163	-1.03163	-1.03163	-1.03163	-1.03163	-1.03163	-1.03163	-1.03163
f_{17}	3.99E-01	4.02E-01	4.05E-01	4.09E-01	3.99E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01
f_{18}	3	3.000001	3.000005	3.000032	3	3	3	3	3
f_{19}	-3.8627	-3.8623	-3.8613	-3.8609	-3.8628	-3.8628	-3.8628	-3.8628	-3.8628
f_{20}	-3.2374	-3.2222	-3.2482	-3.2224	-3.2662	-3.2555	-3.2691	-3.2668	-3.2744
f_{21}	-6.35	-6.64	-6.2	-6.43	-6.1	-5.38	-5.51	-6.8	-6.59
f_{22}	-7.27	-7.84	-7.67	-7.14	-7.43	-6.35	-6.63	-7.5	-6.43
f_{23}	-7.33	-8.53	-8.12	-7.88	-7.68	-6.45	-8.28	-5.92	-6.31

We then apply method A (a=1, b=0.5) to half of the population and the other half uses setting D (a=1, b=2). Note that method A is capable of local search ($a>b$) and method D is capable of global search ($b>a$). The result came to be as expected; 19 out of 23 functions are solved with improvement compared to linearly decreasing w. Nevertheless, for convenience, we recommend the setting of a=1 and b=1 for general purposes. Besides, we also run two additional simulations. The first one involve

choosing either method A or D in a random manner. In the second simulation, we apply a switch from D to A after half of the total generation. Both simulations have the same conclusion as mentioned above with 19 and 18 improvements as compared to the linearly decreasing *w*.

5 Conclusions

In this work, we proposed the exponential inertia weight, *w* to improve the search quality of Particle Swarm Optimization (PSO) algorithm. This exponential *w* has simple mathematical term shown by Eq. (2). The mathematical term originated from the work of Ao and Chi in [17] that is based on the concept of adaptive crossover rate used in Differential Evolution (DE) algorithm. The same formula is adopted and applied to inertia weight, *w*. We further investigate the characteristics of the adaptive *w* graphically and careful analysis showed that there exist two important parameters in the equation for adaptive *w*; one acting as the local attractor, *a* and the other as the global attractor, *b*. We further analyze that the algorithm is capable of global search and local search when ($b>a$) and ($a>b$) respectively. Simulation results showed that the proposed method has better performance in comparison to the linearly decreasing inertia weight that is used widely in many significant works. Among all ten methods, A to J; seven methods (A, B, C, G, H, I, J) managed to obtained better results for 15 and above benchmark problems as compared to linearly decreasing *w*. For convenience, we recommend the setting of both local and global attractors to unity values ($a=b=1.0$). The proposed technique is reliable as 23 benchmark problems are adopted to validate the robustness of the exponential *w*. Further works should investigate and relate information such as convergence, diversity, swarm size, number of dimensions etc. to either local attractor, *a* or global attractor, *b*. Once an effective relationship is found, *a* and *b* will be adjusted automatically and hence resulting in an adaptive *w*. This remains as an important work for future.

References

1. Shi, Y., Eberhart, R.: A Modified Particle Swarm Optimizer. In: IEEE World Congress on Computational Intelligence Evolutionary Computation Proceedings, 1998, pp. 69–73 (1998)
2. Hussain, Z., Noor, M.H.M., Ahmad, K.A., et al.: Evaluation of Spreading Factor Inertial Weight PSO for FLC of FES-Assisted Paraplegic Indoor Rowing Exercise. In: 2011 IEEE 7th International Colloquium on Signal Processing and its Applications (CSPA), pp. 430–434 (2011)
3. Bansal, J.C., Singh, P.K., Saraswat, M., et al.: Inertia Weight Strategies in Particle Swarm Optimization. In: 2011 Third World Congress on Nature and Biologically Inspired Computing (NaBIC), pp. 633–640 (2011)
4. Han, W., Yang, P., Ren, H., et al.: Comparison Study of several Kinds of Inertia Weights for PSO. In: 2010 IEEE International Conference on Progress in Informatics and Computing (PIC), vol. 1, pp. 280–284 (2010)

5. Mekhamer, S.F., Moustafa, Y.G., EI-Sherif, N., et al.: A Modified Particle Swarm Optimizer Applied to the Solution of the Economic Dispatch Problem. In: 2004 International Conference on Electrical, Electronic and Computer Engineering, ICEEC 2004, pp. 725–731 (2004)
6. Zhu, Z., Zhou, J., Ji, Z., et al.: DNA Sequence Compression using Adaptive Particle Swarm Optimization-Based Memetic Algorithm. IEEE Transactions on Evolutionary Computation 15, 643–658 (2011)
7. Seo, J.-H., Im, C.-H., Heo, C.G., et al.: Multimodal Function Optimization Based on Particle Swarm Optimization. IEEE Transactions on Magnetics 42, 1095–1098 (2006)
8. Xin, J., Chen, G., Hai, Y.: A Particle Swarm Optimizer with Multi-Stage Linearly-Decreasing Inertia Weight. In: International Joint Conference on Computational Sciences and Optimization, CSO 2009, vol. 1, pp. 505–508 (2009)
9. Zheng, Y.-L., Ma, L.-H., Zhang, L.-Y., et al.: Empirical Study of Particle Swarm Optimizer with an Increasing Inertia Weight. In: The 2003 Congress on Evolutionary Computation, CEC 2003, vol. 1, pp. 221–226 (2003)
10. Miranda, V., de Magalhaes Carvalho, L., da Rosa, M.A., et al.: Improving Power System Reliability Calculation Efficiency with EPSO Variants. IEEE Transactions on Power Systems 24, 1772–1779 (2009)
11. Feng, Y., Teng, G.-F., Wang, A.-X., et al.: Chaotic Inertia Weight in Particle Swarm Optimization. In: Second International Conference on Innovative Computing, Information and Control, ICICIC 2007, p. 475 (2007)
12. Feng, Y., Teng, G.-F., Wang, A.-X.: Comparing with Chaotic Inertia Weights in Particle Swarm Optimization. In: 2007 International Conference on Machine Learning and Cybernetics, vol. 1, pp. 329–333 (2007)
13. Li, H.-R., Gao, Y.-L.: Particle Swarm Optimization Algorithm with Exponent Decreasing Inertia Weight and Stochastic Mutation. In: Second International Conference on Information and Computing Science, ICIC 2009, vol. 1, pp. 66–69 (2009)
14. Mahor, A., Prasad, V., Rangnekar, S.: Scheduling of Cascaded Hydro Power System: A New Self Adaptive Inertia Weight Particle Swarm Optimization Approach. In: International Conference on Advances in Recent Technologies in Communication and Computing, ARTCom 2009, pp. 565–570 (2009)
15. Zhan, Z.-H., Zhang, J., Li, Y., et al.: Adaptive Particle Swarm Optimization. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 39, 1362–1381 (2009)
16. Dong, C., Wang, G., Chen, Z., et al.: A Method of Self-Adaptive Inertia Weight for PSO. In: 2008 International Conference on Computer Science and Software Engineering, vol. 1, pp. 1195–1198 (2008)
17. Ao, Y., Chi, H.: An Adaptive Differential Evolution to Solve Constrained Optimization Problems in Engineering Design. Scientific Research 2, 65–77 (2010)
18. Yao, X., Liu, Y., Lin, G.: Evolutionary Programming made Faster. IEEE Transactions on Evolutionary Computation 3, 82–102 (1999)