International Journal of Knowledge-based and Intelligent Engineering Systems 15 (2011) 65–70
DOI 10.3233/KES-2010-0211
IOS Press

Comparing nonlinear inertia weights and constriction factors in particle swarm optimization

Yutthapong Tuppadung and Werasak Kurutach*
Faculty of Information Sciences, Nakhonratchasima College, Nakhonratchasima, 30000, Thailand

Abstract. In this paper, we empirically study the performance of particle swarm optimization (PSO) using a nonlinear inertia weight compared with the performance using a constriction factor. Adjusting the parameters of nonlinear inertia weight directly affects the performance of PSO. Five benchmark functions are used for evaluation. Under all testing functions, the PSO with nonlinear inertia weights consistently converges faster than the PSO with constriction factors.

Keywords: Nonlinear inertia weight, particle swarm optimization, constriction factor, PSO, nonlinear decreasing inertia weight

1. Introduction

Particle swarm optimization (PSO) is a population-based, self-adaptive search optimization technique first introduced by Kennedy and Eberhart in 1995 [1]. The development of this method was based on the simulation of simplified animal social behaviors, e.g., fish schooling, bird flocking, etc.

Shi and Eberhart have published several papers that describe research concerning a version of the particle swarm algorithm that incorporates what we call an inertia weight [2,4–6].

Equations (1) and (2) describe the velocity and position update equations with the inertia weight included. Equation (1) calculates a new velocity for each particle (potential solution) based on its previous velocity, the location where the best fitness so far has been achieved, and the population global (or local neighborhood, in the neighborhood version of the algorithm) location where the best fitness so far has been achieved. Equation (2) updates each particle's position in the solution hyperspace. The two random numbers are independently generated. The use of the inertia weight w, which typically decreases linearly from about 0.9 to 0.4 during a run, has provided improved performance in a number of applications.

$$v_{id} = w * v_{id} + c_1 * rand() * (p_{id} - x_{id}) + c_2 * rand() * (p_{gd} - x_{id}) \quad (1)$$

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

In this paper, we propose a nonlinear inertia weight for Eq. (1). Parameters of the nonlinear inertia weight function directly affect PSO performance. Therefore, adjusting parameters should be considered before being used in real world application. The five functions were used to evaluate the performance in comparison with the constriction factor that was introduced by Clerc (1999) [3–6].

The organization of this paper is as follows. In Section 2, we describe the non-linear benchmark functions. Section 3 presents the background concept of the constriction factor. Section 4 introduces the background concept of the nonlinear inertia weight function that is to be applied to PSO. Then, in Section 5, the result of the experimentation is discussed. Finally, we summarize our work in Section 6.

*Corresponding author: Department of Information Science and Technology, Mahanakorn University of Technology, Bangkok 10530, Thailand. E-mail: yutt_t@pea.co.th and werasak@mut.ac.th.

ISSN 1327-2314/11/$27.50

2. Experimental approach

For comparison, five non-linear benchmark functions are used here. The first is Ackley's function described by Eq. (3):

$$f_1(\vec{x}) = -20e^{\left(-0.2\sqrt{\frac{1}{D}\sum_{d=1}^{D} x_d^2}\right)} - e^{\left(\frac{1}{D}\sum_{d=1}^{D}\cos(2\pi x_d)\right)} + 20 + e \tag{3}$$

where $\vec{x} = \{x_1, x_2, \ldots, x_D\}$ is a D-dimensional real-valued vector. The second function is the Alpine function described by Eq. (4):

$$f_2(\vec{x}) = \sum_{d=1}^{D} |x_d \sin(x_d) + 0.1x_d| \tag{4}$$

The third function is the tripod function described by Eq. (5):

$$\begin{aligned} f_3(\vec{x}) &= p(x_2)(1 + p(x_1)) \\ &+ |x_1 + 50p(x_2)(1 - 2p(x_1))| \\ &+ |x_2 + 50(1 - 2p(x_2))| \end{aligned} \tag{5}$$

where $\begin{cases} p(x_i) = 1 & x_i \geqslant 0 \\ = 0 & x_i \leqslant 0 \end{cases}$

The fourth function is the generalized Ratrigrin function described by Eq. (6):

$$f_4(\vec{x}) = \sum_{d=1}^{D} (x_d^2 - 10\cos(2\pi x_d) + 10) \tag{6}$$

The fifth function is Schaffer's f6 function which is described by Eq. (7):

$$f_5(\vec{x}) = 0.5 - \frac{(\sin\sqrt{x_1^2 + x_2^2})^2 - 0.5}{(1.0 + 0.001(x_1^2 + x_2^2))^2} \tag{7}$$

In all cases, the population size was set to 30, the maximum number of iterations was set to 800 and if the GBest was not changed at least 1e-099 for 100 epochs, the program would be terminated.

In all cases for which the inertia weight was used, it was set to 0.9 at the beginning of the run, and made to decrease non-linearly to 0.4 at the maximum number of iterations. Inertia weight cases used a $V_{\max}$ set to the maximum range $X_{\max}$. Each of the two $(p - x)$ terms was multiplied by an acceleration constant of 2.0 (times a random number between 0 and 1).

In all cases for which Clerc's constriction method was used, φ was set to 4.1 and the constant multiplier χ is thus 0.729. This resulted in the previous velocity being multiplied by 0.729, and each of the two $(p - x)$ terms being multiplied by 0.729 * 2.05 = 1.49445 (times a random number between 0 and 1). Note that this is functionally equivalent to using Eq. (1) with $w = 0.729$ and $c_1 = c_2 = 1.49445$. These new parameter values are, however, related (see Eq. (8)).

3. Constriction factor

Since particle swarm optimization originated from efforts to model social systems, a thorough mathematical foundation for the methodology has not been developed at the same time as the algorithm. Within the last few years, a few attempts have been made to begin to establish this foundation. Recent work done by [3] has indicated that the use of a constriction factor may be necessary to ensure convergence of the particle swarm algorithm. A detailed discussion of the constriction factor is beyond the scope of this paper, but a simplified method of incorporating it appears in Eq. (8), where χ is a function of c_1 and c_2 as reflected in Eq. (9) [3,7].

$$\begin{aligned} v_{id}^{t+1} &= \chi(w * v_{id}^t + c_1 * rand() * (p_{id} - x_{id}^t) \\ &+ c_2 * Rand() * (p_{gd} - x_{id}^t)) \end{aligned} \tag{8}$$

$$\chi = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}, \quad \text{where } \varphi = c_1 + c_2, \varphi > 4 \tag{9}$$

Typically, when Clerc's constriction method is used, φ is 4.1 and the constant multiplier χ is 0.729. This results in the previous velocity being multiplied by 0.729 and each of the two $(p - x)$ terms being multiplied by 0.729*2.05 = 1.49445 (times a random number between 0 and 1)

4. Nonlinear inertia weight function

The optimum parameter has been used to prevent the particle position movement achieving an out of velocity rate. The inertia weight control is the influence of previous velocity on the new velocity. Large inertia weight causes larger exploration of the search space, while smaller inertia weights focus the search on a smaller region and should be $w \leqslant 1$. PSO will diverge if this definition is not followed: $w = 0.5\,(c_1 + c_2) - 1$. Typically, PSO is started with a large inertia weight, which is decreased over time. The positive acceleration

constants should be $c_1 + c_2 \leqslant 4$, if $c_1 + c_2 \geqslant 4$ has affected both the velocity and the position divergently to infinite [7].

A nonlinear inertia weight function as Eq. (10) is used to improve the inertia weight because that can adjust the relation between large and small explorations of the space. The characteristic of the inertia weight is controlled by F_1 and F_2 parameters in the formula as follows [8]

$$w(x) = \frac{1}{1 + \left(\frac{x}{F_2}\right)^{F_1}} \tag{10}$$

where $x =$ the number of iterations
$F_1 =$ the speed controller
$F_2 =$ the slope controller. For Example, if $w(40) = 0.5$ when $F_2 = 40$

Figure 1 shows the relation between a characteristic of inertia weight and adjusting parameters(F_1 and F_2). The Y axis is the inertia weight which starts 1 to 0 and the X axis is iteration numbers which start 1 to 100. For example, F_1 is 2 and F_2 are 10(the solid line) and 25 (the dotted line) respectively. The inertia weight speed of the solid line is faster than the dotted line and these are shown in Fig. 1(a). Conversely, in Fig. 1(b), F_1 are 2(the solid line) and 4(the dotted line) respectively, and F_2 is 10. The inertia weight speeds of the solid line and the dotted line are almost the same because F_2 of both lines has been adjusted to the same value ($F_2 =$ 10). Figure 1(c) and (d) also present the characteristic of inertia weight when F_2 are 25 and 40 respectively, and F_1 are still 2 and 4 respectively.

5. Experiment and results

The performance comparison between nonlinear inertia weights and constriction factors is presented by the number of iterations. This evaluates a performance of convergence based on the average number of iterations and the range values of iteration. The experiment applied the Particle Swarm Optimization toolbox (PSOt) for MATLAB which is programmed by Brian Birge. In our experiment, each version of each approach (nonlinear inertia weight and constriction factor) was run 10 times for each test function.

Section 4 describes the main concept of setting the optimum parameter for preventing the particle position movement achieving an out of velocity rate. The nonlinear parameters F_1 and F_2 are 4 and 80 iterations (10% of maximum iterations) respectively, as defined in this paper. We are seeking to define the velocity of PSO in pursuing the global objective within the first 10% of iterations (80 iterations), with F_1 as 4. If F_1 is a large value, the velocity will decrease too slowly. We set F_2 as 80 iterations or 10% of iteration for pursuit of the local objective after 80 iterations.

Table 1
Ackley's function iterations using nonlinear inertia weight

250	178	105	105	162
208	109	174	108	181

Table 2
Ackley's function iterations using constriction factor

103	305	300	305	311
320	313	309	307	310

The benefit of parameters setting is that the global objective is explored rapidly while the local objective is sought slowly, and convergence to the target is achieved fast and correctly. The nonlinear inertia weight parameter setting is used in a real world application as [9] which defines F_1 is 4 and F_2 is 10% of maximum iterations. As a result, the PSO algorithm converges to the target faster than the other parameter as presented in Section 4.

5.1. Ackley's function

The dimensions and generation of Ackley's function are 30 and 800, respectively. The program would be terminated when an error of less than 1e-099 for 100 epochs was achieved. Table 1 shows the number of iterations which converge to the target by the nonlinear inertia weight method; Table 2 shows iterations needed using the constriction factor method.

The average number of iterations using the nonlinear inertia weight is thus 158; the range of values is 145 (from 105 to 250), about 91.7 percent of the average, and standard deviation is 50.2.

The average number of iterations using the constriction factor is thus 288.3; the range of values is 217 (from 103 to 320), about 75.3 percent of the average, and standard deviation is 65.32.

As a result, the nonlinear inertia weight method achieves convergence on the target faster than the constriction factor method, evaluated by the average number of iterations and the range of values. If testing of the performance is based on the processing time, the results of both methods are not different.

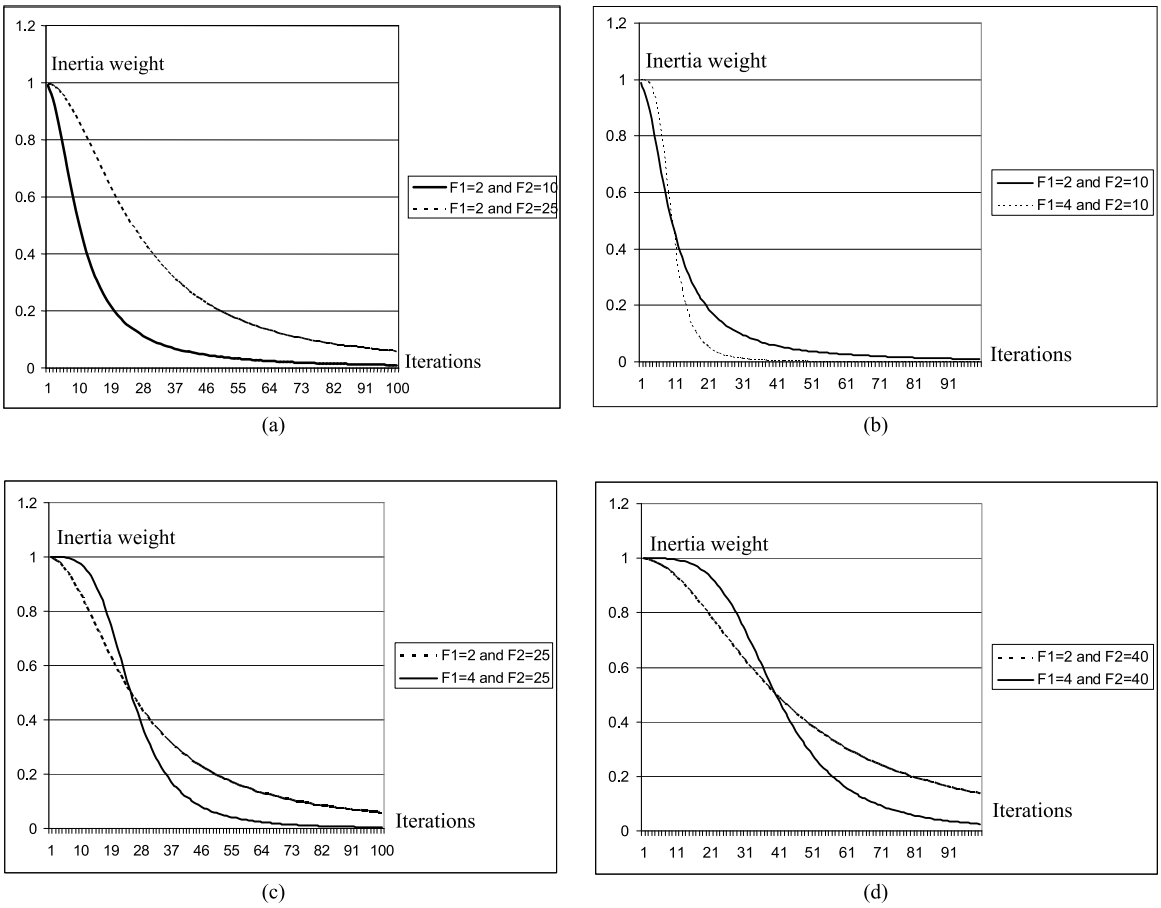

Fig. 1. The characteristic of inertia weight will be changed when the F_1 and F_2 parameters are adjusted.

Table 3
Alpine function iterations using nonlinear inertia weight

104	139	101	104	288
105	108	105	105	158

Table 4
Alpine function iterations using constriction factor

800	800	800	800	800
800	800	800	800	800

5.2. Alpine function

The dimensions and generation of the Alpine function are 30 and 800, respectively. The program would be terminated when an error of less than 1e-099 for 100 epochs was achieved. Table 3 shows the number of iterations which converge to the target by the nonlinear inertia weight method; Table 4 shows iterations needed using the constriction factor method.

The average number of iterations using the nonlinear inertia weight is thus 131.7; the range of values is 187 (from 101 to 288), about 142 percent of the average, and standard deviation is 58.07.

The target error was not achieved within 800 iterations for all runs.

As a result, the constriction factor method is not able to converge to the target of the function within 800 iterations but the nonlinear inertia weight method converges to the target quickly. If testing of the performance is based on the processing time, the results of both methods are not different.

5.3. Tripod function

The dimensions and generation of the Tripod function are 30 and 800, respectively. The program would be terminated when an error of less than 1e-099 for 100 epochs was achieved. Table 5 shows the number of iterations which converge to the target by the nonlinear

Table 5
Tripod function iterations using nonlinear inertia weight

193	193	117	196	215
105	105	114	265	112

Table 6
Tripod function iterations using constriction factor

314	323	316	320	321
314	308	320	317	293

Table 7
Ratrigrin function iterations using nonlinear inertia weight

284	105	104	119	116
130	153	104	196	111

inertia weight method; Table 6 shows iterations needed using the constriction factor method.

The average number of iterations using the nonlinear inertia weight is thus 161.5; the range of values is 160 (from 105 to 265), about 99 percent of the average, and standard deviation is 57.56.

The average number of iterations using the constriction factor is thus 214.6; the range of values is 28 (from 293 to 321), about 8.9 percent of the average, and standard deviation is 8.74.

As a result, the performance of the nonlinear inertia weight method achieves convergence to the target faster than the constriction factor method, evaluated by the average number of iterations and the range of values. If testing of the performance is based on the processing time, the results of both methods are not different.

5.4. Ratrigrin function

The dimensions and generation of the Ratrigrin function are 30 and 800, respectively. The program would be terminated when an error of less than 1e-099 for 100 epochs was achieved. Table 7 shows the number of iterations which converge to the target by the nonlinear inertia weight method; Table 8 shows iterations needed using the constriction factor method.

The average number of iterations using the nonlinear inertia weight is thus 142.2; the range of values is 180 (from 104 to 284), about 126.6 percent of the average, and standard deviation is 57.51.

The average number of iterations using the constriction factor is thus 214.1; the range of values is 232 (from 105 to 337), about 102 percent of the average, and standard deviation is 58.76.

Table 8
Ratrigrin function iterations using constriction factor

232	230	105	237	222
222	225	229	220	219

Table 9
Schaffer's f6 function iterations using nonlinear inertia weight

167	188	145	151	193
326	163	220	226	139

Table 10
Schaffer's f6 function iterations using constriction factor

299	233	218	276	234
264	215	304	205	266

As a result, the performance of the nonlinear inertia weight method achieves convergence to the target faster than the constriction factor method, evaluated by the average number of iterations and the range of values. If testing of the performance is based on the processing time, the results of both methods are not different.

5.5. Schaffer's f6 function

The dimensions and generation of the Schaffer's f6 function are 30 and 800, respectively. The program would be terminated when an error of less than 1e-099 for 100 epochs was achieved. Table 9 shows the number of iterations which converge to the target by the nonlinear inertia weight method; Table 10 shows iterations needed using the constriction factor method.

The average number of iterations using the nonlinear inertia weight is thus 191.8; the range of values is 187 (from 139 to 326), about 97.5 percent of the average, and standard deviation is 55.85.

The average number of iterations using the constriction factor is thus 251.4; the range of values is 99 (from 205 to 304), about 39.4 percent of the average, and standard deviation is 35.34.

As a result, the performance of the nonlinear inertia weight method achieves convergence to the target faster than the constriction factor method, evaluated by the average number of iterations and the range of values. If testing of the performance is based on the processing time, the results of both methods are not different.

6. Conclusion

In this paper, the performances of the PSO algorithm with nonlinear decreasing inertia weight and constric-

tion factor are evaluated by experimental studies of five non-linear functions. The experiment results show that the PSO algorithm with nonlinear decreasing inertia weight has the ability to converge faster than the PSO algorithm with constriction factor.

The results also show not only that the average number of iterations in each function that applies constriction factor were more than with decreasing inertia weight, but also that the target error was not achieved within 800 iterations for all runs in the Alpine function. Applying the nonlinear inertia weight function to PSO improves its performance. Adjusting the parameters directly affects, and can benefit, evaluation in the search space. Therefore, before real world application of the PSO with nonlinear decreasing inertia weight, nonlinear inertia weight parameters should be selected with care.

References

[1] J. Kennedy and R.C. Eberhart, Particle swarm optimization, Proc. IEEE International Conference on Neural Networks (Perth, Australia), *IEEE Service Center*, Piscataway, NJ, 1995, IV: 1942–1948.

[2] Y. Shi and R. Eberhart, Parameter selection in particle swarm optimization, In Evolutionary Programming VIZ: Proc. EP98, New York: Springer Verlag, 1998, pp. 591–600.

[3] M. Clerc, The swarm and the queen: towards a deterministic and adaptive particle swarm optimization, Proc. I999 ICEC, Washington, DC, 1999, pp. 1951–1957.

[4] Y. Shi and R.C. Eberhart, Empirical Study of Particle Swarm Optimization, *CEC* **99** (1999), 1945–1150.

[5] R.C. Eberhart and Y. Shi, Comparing inertia weights and constriction factors in particle swarm optimization, *Congress on Evolutionary Computation* (2000), 85–88.

[6] R.C. Eberhart and Y. Shi, Particle swarm optimization: developments, applications and resources, *Congress on Evolutionary Computation* (2001), 81–86.

[7] P. Engelbrecht, Computational Intelligence, West Sussex: Jonh Wiley and Sons, 2002.

[8] T.J. Ross, Fuzzy Logic with Engineering Application, McGraw-Hill, Inc, New York, 1995.

[9] Y. Tuppadung and K. Kurutach, Modified Particle Swarm Optimization for an Optimal Feeder-Switch Relocation, *International Journal on Artificial Intelligence Tools* **17**(2) (Apr. 2008).