EMITTER International Journal of Engineering Technology
Vol.3, No.2, December 2015
ISSN: 2443-1168

Modified Particle Swarm Optimization using Nonlinear Decreased Inertia Weight

Alrijadjis[1], Shenglin Mu[2], Shota Nakashima[3], Kanya Tanaka[4]

[1)]Politeknik Elektronika Negeri Surabaya, [2)]Hiroshima National College of Maritime Technology, [3),4)]Yamaguchi University
[1)]Jl. Raya ITS, Keputih, Sukolilo, Surabaya, Indonesia, Telp. +62(31)5947280, [2)]4272-1 Higashino, Osakikamijima, Toyota District, Hiroshima 725-0231, Japan, Telp. +81-846-65-3101, [3),4)]2-16-1 Tokiwadai, Ube-shi 755-8611, Japan, Telp. +81-836-85-9005
E-mail: [1)] alrijadjis@pens.ac.id, [2)] mshenglin@hiroshima-cmt.ac.jp,
[3)] s-naka@yamaguchi-u.ac.jp, [4)] ktanaka@yamaguchi-u.ac.jp

Abstract

Particle Swarm Optimization (PSO) has demonstrated great performance in various optimization problems. However, PSO has weaknesses, namely premature convergence and easy to get stuck or fall into local optima for complex multimodal problems. One of the causes of these weaknesses is unbalance between exploration and exploitation ability in PSO. This paper proposes a Modified Particle Swarm Optimization (MPSO) using nonlinearly decreased inertia weight called MPSO-NDW to improve the balance. The key idea of the proposed method is to control the period and decreasing rate of exploration-exploitation ability. The investigation with three famous benchmark functions shows that the accuracy, success rate, and convergence speed of the proposed MPSO-NDW is better than the common used PSO with linearly decreased inertia weight or called PSO-LDW

Keywords: particle swarm optimization (PSO), premature convergence, local optima, exploration ability, exploitation ability.

1. INTRODUCTION

The difficulties associated with using mathematical optimization on large-scale complex engineering problem have contributed to the development of alternative solution. To overcome these problems, researchers proposed evolutionary-based algorithm for searching near-optimum solutions to problems. Evolutionary algorithms are stochastic search methods that mimic the metaphor of natural biological evolution and/or the social behavior or species. To mimic the efficient behavior of these species, various researchers have developed computational systems that seek fast and robust solutions to complex optimization problems.

Particle Swarm Optimization (PSO) is one of the evolutionary computational technique developed by Kennedy and Eberhart in 1995 [1]. It is a population-based stochastic search algorithm inspired by the simulation of the behavior of the birds flocking or fish schooling. The basic idea of PSO comes from the research of the behavior for the bird swarm to catch food. PSO has shown good performance in finding good solution to optimization problems, and turned out to be another powerful tool besides other evolutionary algorithm such as Genetic Algorithm (GA) [2], [3]. Compared with genetic algorithm and ant algorithm, PSO has simple algorithm or form, faster convergence, efficient in time-calculation and is easily implemented as well as the adjustable parameters are few, so PSO is adept to solving many non-derivative and multi-peak complex optimization problems. PSO has been successfully applied to many science and practical fields [4]-[7].

Although PSO has superior features, it has some problems, such as premature convergence and fall into local optima [2], [8]. It was reported that the causes of the problem are unbalance between exploration-exploitation ability and lost-diversity or lack-information due to fast rate flow in sharing information. Exploration ability or global search ability is the ability to identify a region with a best solution. Particles with strong exploration ability have a high speed velocity to search in a wider area. Exploitation ability or local search ability is the ability to find a best solution in a targeted area or limited area. Particles with strong exploitation ability have low speed velocity to refine and capture a best solution. If the particles are far from a best solution, strong exploration ability is better. If the particles are close to a best solution, strong exploitation ability is better. Due to random process and particle's movement, the position of particles is always changed in each iteration. So, the control of both abilities in order to get a proper balance is needed. In previous method called PSO using linearly decreased inertia weight (PSO-LDW) [9], inertia weight adjustment was used to control these abilities. In the first iteration, inertia weight is set in maximum value, and then it is linearly decreased until minimum value at the end iteration. Although, PSO-LDW is better than PSO using constant inertia weight, sometimes it suffers from the problem of being trapped in local optima and premature convergence.

In this paper, the PSO-LDW is revised with a nonlinearly decreased inertia weight, proposed to efficiently control the period and decreasing rate of exploration-exploitation ability. A new parameter called nonlinear index number is added to control the path of inertia weight. The proposed MPSO-NDW was verified on three benchmark functions and the results were compared with the original PSO-LDW.

The rest of the paper is organized as follows. In section 2, the original PSO is introduced. In section 3, the modified PSO with nonlinearly decreased inertia weight is proposed. In section 4, experiments on several benchmark functions are done to test our proposed method, then we compare it with the original PSO, and the simulation result is analysed. Finally, section 5 concludes with a summary.

2. PARTICLE SWARM OPTIMIZATION ALGORITHM

PSO is a population-based optimization method using the concept of cooperation inspired by the behavior of organism, such as birds flocking or fish schooling, in search for food [1]. The outline for PSO is marked as follows. Let consider the optimization problem of maximizing the evaluation function $f : M \rightarrow M' \subset R$ for variable $x \in M \subset R^n$. Let there be N particles (mass point) on M dimensional space, where the position vector and velocity vector of $i(= 1,2,3,....,N)$th particle for m searching number arex_i^m and v_i^m . The best position for each particle in the evaluation function $f(x)$ of searching point is represented as Pb_i (*Pbest*), while the best position of $f(x)$ in the searching point for the whole particle is represented as gb (*gbest*). The particles are manipulated according to the following recurrence equations:

$$v_i^{m+1} = w \cdot v_i^m + c_1 \cdot r_1 \cdot (Pb_i - x_i^m) + c_2 \cdot r_2 \cdot (gb_i - x_i^m) \quad (1)$$

$$x_i^{m+1} = x_i^m + v_i^{m+1} \quad (2)$$

$$w = w_{max} - (w_{max} - w_{min}) \cdot \frac{m}{m_{max}} \quad (3)$$

where$w$ is the inertia weight; c_1 and c_2 are cognitive and social constant; r_1 and r_2 are random numbers. There are three parts or vectors that affect the particle's movement, i.e., momentum vector, ($w.v$), cognitive vector, (Pb – x), and social vector, (gb –x). According to Eq. (1) and Eq. (2), the particle's movement in PSO can be illustrated in Fig. 1. The next position of particle is the resultant of three vectors.

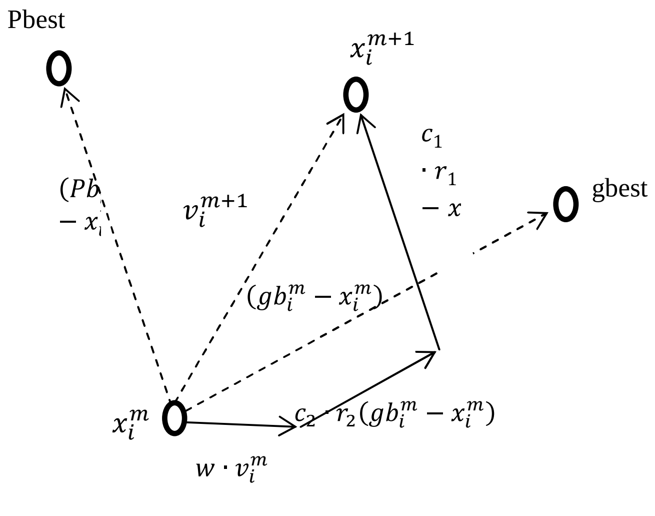

Figure 1. Particle's movement in PSO

The working mechanism of PSO algorithm can be described in four steps as follows:

1. Deploy a population of candidate solution (or particles) in the searching-area randomly. Each particle can handle a candidate solution with D-dimension.
2. Evaluate the fitness value of each particle and set as pbest and gbest.
3. Update the position and velocity of each particle using Eq. 1 and Eq. 2.
4. Check the termination condition. If the condition is not met, return to No. 2. If the condition is met, the process is complete and the optimal solution is the particle with gbest.

The flowchart of the PSO algorithm is shown in Fig. 2.

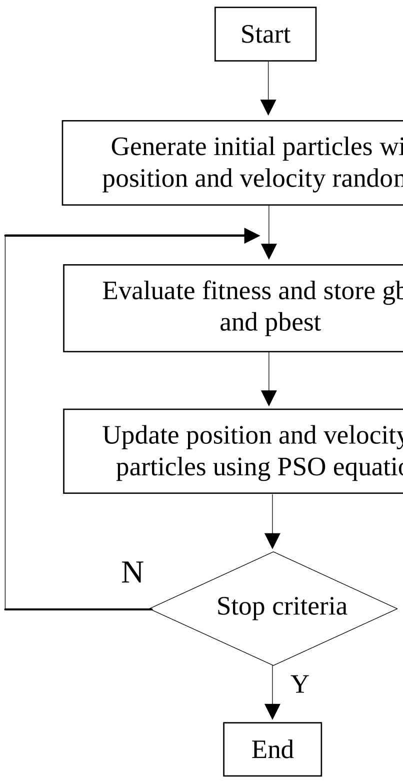
Start
Generate initial particles with position and velocity randomly
Evaluate fitness and store gbest and pbest
Update position and velocity of particles using PSO equation
N
Stop criteria
Y
End

Figure 2. The flowchart of PSO

3. PROPOSED MODIFIED PSO

The most important parameter of PSO is inertia weight because of its capability to control the balance of exploration-exploitation abilities. Recently, research to improve PSO is being conducted intensively. Improving PSO is focused on how to adjust inertia weight in order to get a proper balance. The relationship between inertia weight and ability in PSO is shown in Fig. 3. The range of inertia weight is from 0.1 to 0.9. The lower inertia weight will cause strong exploitation ability (β) and weak exploration ability (α). The higher inertia weight will cause weak exploitation ability and strong exploration ability. The combination of both abilities is one ($\alpha + \beta = 1$). The problem is how to adjust inertia weight in order to get a proper combination or balance between exploration-exploitation abilities.

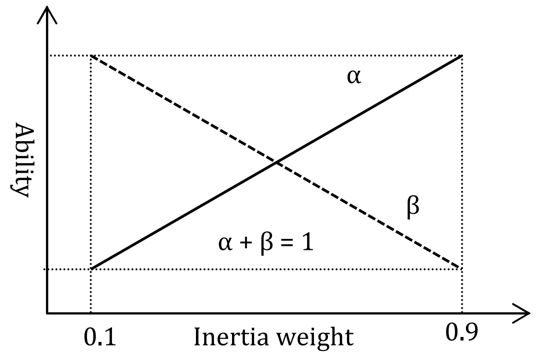

Figure 3. Inertia weight and abilities of PSO

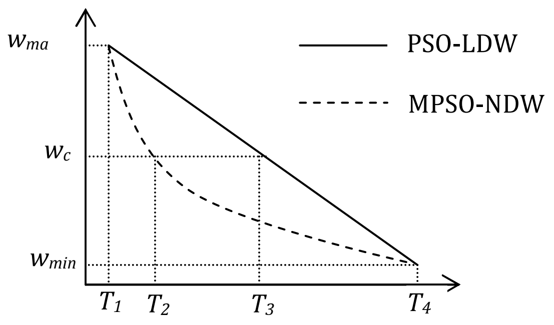

Figure 4. Path of inertia weight

In the original PSO, inertia weight is linearly decreased from maximum value to minimum value as shown in Eq. 3. It is called PSO-LDW [8]. As shown in Fig. 4, due to this strategy, the usage period of exploration ability ($T_\alpha = T_3 - T_1$) is similar with the usage period of exploitation ability ($T_\beta = T_4 - T_3$), or $T_\alpha = T_\beta$. Also, the decreasing rate of exploration ability ($\Delta_\alpha = \frac{w_{max}-w_c}{T_3-T_1}$) is similar with the increasing rate of exploitation ability ($\Delta_\beta = \frac{w_c-w_{min}}{T_4-T_3}$), or $\Delta_\alpha = \Delta_\beta$. So, there is no controlling of the usage period and changing rate of exploration-exploitation abilities in PSO-LDW. It is a weakness of PSO-LDW.

In the proposed method, we want to investigate the impact of usage period and decreasing or increasing rate of exploration and exploitation ability. To control the period and the changing rate, the path of inertia weight must be made nonlinear using a nonlinear index number. It means that inertia weight is nonlinearly decreased from maximum value to minimum value by the following equation:

$$\mathrm{w} = w_{min} + (w_{max} - w_{min}) \cdot \left(\frac{m_{max}-m}{m_{max}-1}\right)^x \tag{4}$$

where*x* is nonlinear index number. Higher of *x* will cause the usage period of exploration ability is shorter than the usage period of exploitation ability, $T_\alpha <$ T_β, and the decreasing rate of exploration ability is faster than the increasing rate of exploitation ability, $\Delta_\alpha >$

4. EXPERIMENTAL RESULTS

This section compares the performance of the proposed MPSO-NDW with the original or common PSO-LDW discussed in Section 2. To verify and evaluate the efficiency and the effectiveness of the proposed approach we have used three widely known benchmark functions with different characteristics, i.e., Sphere function (with single optimum solution), Rosenbrock's function (with one local optimum and one global optimum) and Griwank's function (with one global optimum and many local optimums), as follows:

$$f_1(x,y) = (x-15)^2 + (y-20)^2 \quad (5)$$

$$f_2(x,y) = 10 \cdot (x^2 - y)^2 + (1-x)^2 \quad (6)$$

$$f_3(x,y) = 1 + \frac{x^2+y^2}{40} - \cos(x) \cdot cos\left(\frac{y}{\sqrt{2}}\right) \quad (7)$$

The global best solution for the Sphere function is zero which is achieved when *x* = 15 and *y* = 20; for the Rosenbrock's function is zero which is achieved when *x* = 1 and *y* = 1; and for the Griwangk's function is zero which is achieved when *x* = 0 and *y* = 0.

For the purpose of comparison, all the simulation deploy the same parameter settings in both of PSO (original PSO-LDW and MPSO-NDW) such as the maximum number of iterations, *itermax* = 20; cognitive constant, c_1 = 1.0; social constant, c_2 = 1.0; number of particles, *N* = 5, maximum inertia weight, w_{max} = 0.9; and minimum inertia weight, w_{min} = 0.1. Since PSO is a stochastic algorithm that randomly searches the best solution, so for testing we have done as much as 100 runs.

Experimental results of MPSO-NDW using different nonlinear index number and PSO-LDW for Sphere function, Rosenbrock function and Griwank function averaged over 100 runs are recorded in Table 1-3, respectively. MPSO-NWD using *x* = 1 is similar with PSO-LDW. By looking at mean error, maximum error, minimum error and standard deviation error, it is easy to see that MPSO-NDW using *x* = 1.2 shows a better accuracy than the other nonlinear index numbers. Due to *x* = 2, the period of exploration ability becomes a little shorter and the period of exploitation ability becomes a little longer. Also, the decreasing rate of exploration ability becomes a little faster and the decreasing rate of exploitation ability becomes a little slower. Since Rosenbrock function has a little difference between global optima and local optima, the particles in both PSO-LDW and MPSO-NDW having difficulty in finding a best solution. Accordingly, the results aren't accurate. From these Tables, it is clearly obvious that a proper controlling of the period and decreasing rate of exploration-exploitation ability gives a good impact for increasing accuracy.

Table 1. Statistical analysis of MPSO-NDW and PSO-LDW for Sphere function

Methods		Mean error	Max error	Min error	Std error
PSO-LDW		0.8786	22.7055	0	3.8032
MPSO-NDW	*x* = 1	0.8786	22.7055	0	3.8032
	x = 1.2	0.3186	10.7007	0	1.4781
	x = 1.5	1.4108	87.096	0	8.9412
	x = 2	3.0093	43.0417	0	8.7923
	x = 5	20.062	124.726	0	27.788
	x = 8	47.332	159.786	0.0001	42.401

Table 2. Statistical analysis of MPSO-NDW and PSO-LDW for Rosenbrock function

Methods		Mean error	Max error	Min error	Std error
PSO-LDW		6.3063	139.355	0.0013	16.279
MPSO-NDW	*x* = 1	6.3063	139.355	0.0013	16.279
	x = 1.2	4.1851	20.734	0.0004	5.4643
	x = 1.5	6.7757	114.82	0.0008	13.838
	x = 2	11.861	275.59	0.0008	39.862
	x = 5	36.151	331.05	0	53.289
	x = 8	41.129	932.93	0.0012	124.01

Table 3. Statistical analysis of MPSO-NDW and PSO-LDW for Griwank function

Methods		Mean error	Max error	Min error	Std error
PSO-LDW		0.2628	2.5566	3.67e-6	0.4152
MPSO-NDW	*x* = 1	0.2628	2.5566	3.67e-6	0.4152
	x = 1.2	0.1837	0.9408	3.91e-6	0.3117
	x = 1.5	0.3401	1.8014	2.24e-7	0.4012
	x = 2	0.4002	2.7190	1.42e-7	0.5103
	x = 5	0.7477	2.7185	5.42e-9	0.7417
	x = 8	0.9926	2.7213	1.38e-6	0.8847

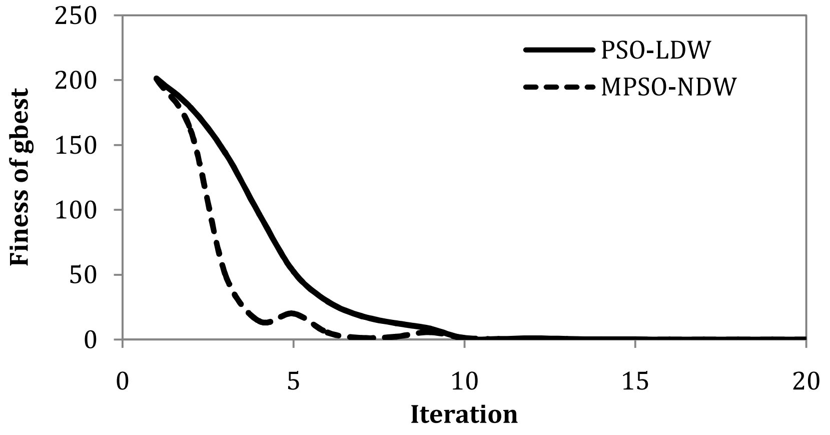

Figure 5. Convergence speed of MPSO-NDW and PSO-LDW for Sphere function

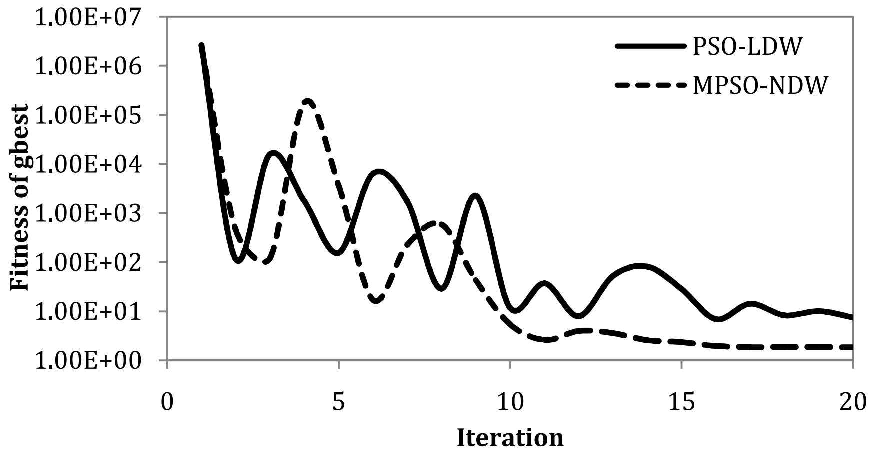

Figure 6. Convergence speed of MPSO-NDW and PSO-LDW forRosenbrock function

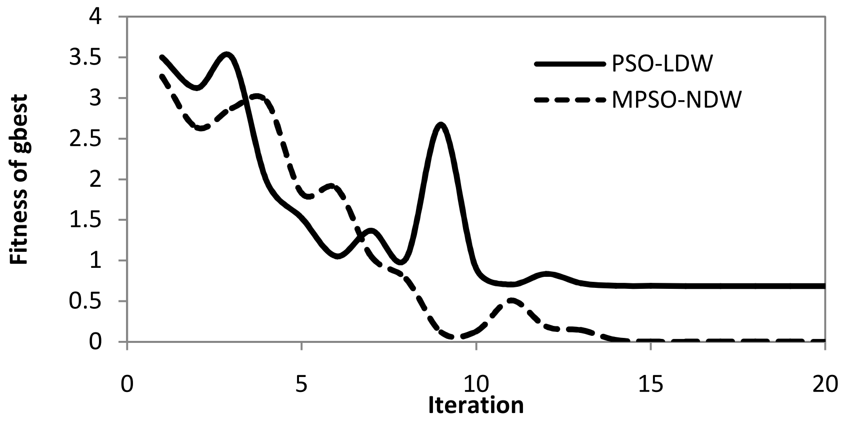

Figure 7. Convergence speed of MPSO-NDW and PSO-LDW forGriwank function

Table 4. Convergence speed of MPSO-NDW and PSO-LDW

Method	Convergence Speed		
	Sphere Function (fit ≤ 0.001)	Rosenbrock Function (fit ≤ 1)	Griwank Function (fit ≤ 0.1)
PSO-LDW	14	18	16
MPSO-NDW	12	16	14

Table 5. Success rate of MPSO-NDW and PSO-LDW

Method	Success Rate		
	Sphere Function (Err ≤ 0.001)	Rosenbrock Function (Err ≤ 0.1)	Griwank Function (Err ≤ 0.001)
PSO-LDW	49	13	37
MPSO-NDW	61	21	47

The comparison of convergence speed between MPSO-NDW and PSO-LDW for Sphere function, Rosenbrock function and Griwank function are shown in Fig. 5, Fig.6, and Fig.7, respectively. It is easy to see that the convergence speed of MPSO-NDW is faster than that of PSO-LDW. Also, Table 4 shows convergence speed averaged over 100 runs in obtaining the predetermined fitness value. In here, the predetermined fitness value for Sphere function, Rosenbrock function and Griwank are 0.001, 1 and 0.1, respectively. To achieve these values, PSO-LDW needed 14, 18, and 16 iterations. While iterations required by MPSO-NDW to achieve these values are 12, 16 and 14, respectively.

Success rate represents the success of method in obtaining a predetermined minimum error within all runs. Table 5 shows the success rate of MPSO-NDW and PSO-LDW within 100 runs for each function. The predetermined minimum error for Sphere function, Rosenbrock function and Griwank function are 0.001, 0.1, and 0.001, respectively. In general, the success rate of MPSO-NDW is higher than PSO-LDW for all benchmark function.

5. CONCLUSIONS

The paper has investigated the impact of the period and decreasing rate of exploration-exploitation ability in PSO. In the original PSO called PSO-LDW, the period and decreasing rate of both abilities was set equal. The proposed MPSO-NDW introduces a new parameter called nonlinear index number to control the period and decreasing rate of both abilities or path of inertia weight. The main contribution of this paper is to show that the nonlinear path of inertia weight can affect the performance of PSO. The second contribution is to give us new ideas for analysis PSO using controlling the usage period of exploration-exploitation abilities in PSO.The proposed MPSO-NDW was applied to three well known benchmark function and compared with the common PSO-LDW. Experimental results indicate that a proper controlling of the period and decreasing rate can increase the performance of PSO in term of accuracy, success rate and convergence speed.

REFERENCES

[1] J. Kennedy, R.C. Eberhart, **Particle Swarm Optimization**, Proceeding IEEE International Conference on Neural Networks, pp. 1942-1945, 1995

[2] Y. Shi, R.C. Eberhart, **A modified particle swarm optimizer**, Proceeding of IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, pp. 69-73, 1998

[3] J. Kennedy, R.C. Eberhart, Y. Shi, **Swarm Intelligence**, San Fransisco: Morgan Kaufman Pubhisher, 2001

[4] A. Chander, A. Chatterjee, P. Siarry, **A new social and momentum component adaptive PSO algorithm for image segmentation**, Expert System with application, No. 38, pp. 4998-5004, 2011

[5] C.L. Chen, R.M. Jan, T.Y. Lee, C.H. Chen, **A novel particle swarm optimization algorithm solution of economic dispatch with valve point loading**, Journal of Marine Science and Technology, Vol. 19, No. 1, pp. 43-51, 2011

[6] H. Zhu, C. Zheng, X. Hu, X. Li, **Adaptive PSO using random inertia weight and its application in UAV path planning**, Proceeding of SPIE, Vol. 7128, pp. 1-5, 2008

[7] K. Tanaka, Y. Murata, Y. Nishimura, Faridah A. Rahman, M. Oka, A. Uchibori. **Variable gain type-PID control using PSO for ultrasonic motor**. Journal of the Japan Society of Applied Electromagnetics and Mechanics, Vol. 18, No. 3, 118-123, 2011

[8] I.C. Trelea, **The particle swarm optimization algorithm: convergence analysis and parameters setting**, Information Processing Letters, No. 85, pp. 317-325, 2003

[9] Y. Shi, R. Eberhart, **Empirical study of particle swarm optimization**, Proceeding of the IEEE Congress on Evolutionary Computation, pp. 1945-1950, 1999