Analytical and Empirical Study of Particle Swarm Optimization with a Sigmoid Decreasing Inertia Weight

Andi Adriansyah[1], **Shamsudin H. M. Amin**[2]

[1]*Department of Electrical, Faculty of Industrial Engineering,*
Universitas Mercu Buana, Jl.Raya Meruya Selatan, Kembangan, Jakarta, 11650, Indonesia
Phone: +6221-5840816, Fax : +6221-5840813, E-mail: aandi2@siswa.utm.my

[2]*Centre of Artificial Intelligence and Robotics (CAIRO), Faculty of Electrical Engineering,*
Universiti Teknologi Malaysia, Skudai, 81310, Johor Bahru, Malaysia
Phone: +607-5535319, Fax: +607-5566272, E-mail: sham@fke.utm.my

Abstract

The particle swarm optimization (PSO) is an algorithm for finding optimal regions of complex search space through interaction of individuals in a population of particles. Search is conducted by moving particles in the space. Some methods area attempted to improve performance of PSO since is founded, including linearly decreasing inertia weight. The present paper proposes a new variation of PSO model where inertia weight is sigmoid decreasing, called as Sigmoid Decreasing Inertia Weight. Performances of the PSO with a SDIW are studied analytically and empirically. The exploration–exploitation tradeoff is discussed and illustrated, as well. Four different benchmark functions with asymmetric initial range settings are selected as testing functions. The experimental results illustrate the advantage of SDIW that may improve PSO performance significantly.

Keywords:

Particle Swarm Optimization; Inertia Weight; Sigmoid Decreasing Inertia Weight

1. Introduction

The difficulties associated with using mathematical optimization on large-scale engineering problem have contributed to the development of alternative solutions. To overcome these problems, researchers have proposed evolutionary-based algorithms 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 evolutionary computation technique developed by Kennedy and Eberhart in 1995 [1, 2]. The method finds the optimal solution by simulating such social behavior of groups as fish schooling or bird flocking. A group can achieve the objective effectively by using the common information of every particle, and the information owned by the particle itself.

However, the PSO algorithm includes some tuning parameters that greatly influence the algorithm performance, known as the exploration-exploitation tradeoff. Balancing between exploration and exploitation searching process will improve PSO performance. A number of methods have been provided to get to the bottom of the problem. Early experience with PSO was proposed by Shi and Eberhart that introduced inertia weight and maximal velocity which tuned based on trial and error [3]. Suitable selection of the inertia weight provides a balance between global and local searching. Afterward, they presented a new concept about inertia weight [4]. In this concept, they attempted to get better of PSO performance by linearly decreasing inertia weight (LDIW). They also tried to overcome the problem by changing inertia weight adaptively based on Fuzzy System [5] and randomly [6]. Furthermore, recent work done by Clerc [7] indicates that use of a constriction factor may be necessary to insure convergence of the PSO. In constriction factor, inertia weight adjusted concurrently with another PSO parameters. In contrast, Zheng et. al., investigated increasing inertia weight in their research [8]. According to them, a PSO with increasing inertia weight outperforms the one with decreasing inertia weight. Though, the results still not satisfied.

This paper presents an approach to overcome exploration-exploitation tradeoff problem. A new nonlinear function modulated inertia weight adaptation with time proposed for improved performance of PSO algorithm. Instead of linearly decreasing inertia weight, the schema attempted to decrease inertia weight by means of sigmoid function. In this work, some analytical and empirical studies are investigated. In section 2, philosophy and procedure of original PSO are explained. Some analysis also presented in this section. In section 3, a new PSO model with a sigmoid decreasing inertia weight (SDIW) is suggested. To prove the validity of such method, several standard benchmark functions are tested in Section 4. The empirical data resulted will be emphasized and discussed in Section 5. Finally Section 6 concludes this paper.

2. PSO Algorithm and Analysis

2.1. Philosophy and Procedure of Standard PSO

PSO is one of the artificial life or multiple particles' type techniques designed and developed by Kennedy and Eberhart [1, 2]. The concept of original PSO can be described as follows: each potential solution, called *particle*, knows its best value so far (*pbest*) and its position. Moreover, each particle knows the best value in the group (*gbest*) among the *pbest*. All of the best values are based on fitness function (*F(.)*) for each problem to be solved. Each particle tries to modify its position using the current velocity and its position. The velocity of each particle can be calculated using the following Equation:

$$v_i^{k+1} = v_i^k + c_1 rand(\,)(pbest \quad s_i^k) + c_2 rand(\,)(gbest \quad s_i^k) \qquad (1)$$

where v_i^k, v_i^{k+i} , and s_i^k , are velocity vector, modified velocity and positioning vector of particle i at generation k, respectively. Then, *pbest* and *gbest* are best position found by particle i and best position found by particle group. Finally, c_1 and c_2 are cognitive and social coefficients, respectively, that influence particles velocity. Afterward, the current position of a particle is calculated by the following Equation:

$$s_i^{k+1} = s_i^k + v_i^{k+1} \qquad (2)$$

Updating process of velocity and position of each particle is depicted in Fig. 1.

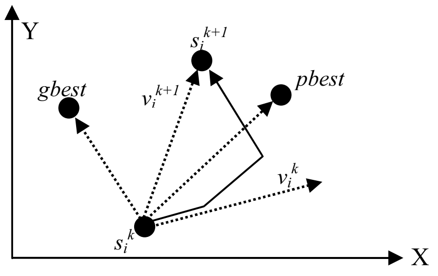

Figure 1. The velocity and position updates in PSO

An algorithm to find the best positioning vector of PSO using n particles can be summarized as follows:

1. Initial positioning vector $S[n]$ and velocity vector $V[n]$ are generated by using random values, where $s_i = [s_i^1, s_i^2, ..., s_i^n]$ and $v_i = [v_i^1, v_i^2, ..., v_i^n]$.
2. Velocity vector v_i^{k+i} of particle i is calculated by using Equation (1).
3. New positioning vector s_i^{k+1} of particle i is calculated by using Equation (2).
4. If F(s_i^k) is better than the F($pbest_i$), the positioning vector s_i^k is set to *pbest*. If F($pbest_i$) is better than F(*gbest*), the positioning vector *gbest* is set to *pbest*.
5. If the generation reaches to the pre-determined one, process will stop. Otherwise, will go to step 2.

In order to get better control exploration and exploitation of particles searched, the concept of inertia weight, w, is developed [3, 4]. Introducing inertia weight concept, Equation (1) and Equation (2) can be written as:

$$v_i^{k+1} = w_i v_i^k + c_1 rand(\,)(pbest \quad s_i^k) + c_2 rand(\,)(gbest \quad s_i^k) \qquad (3)$$

$$s_i^{k+1} = s_i^k + v_i^{k+1} \qquad (4)$$

Almost previous researches on PSO system have provided empirical results and informal analyses. This paper presents formal analysis of traditional simple particle systems, crucial to understand the dynamics of how particle behavior depends on parameters for making the right choice of parameter values.

2.2. Particle Trajectory Analysis

It appears in Equation. (3) and (4) that each dimension is updated independently from the others. The only link between the dimensions of the problem space is introduced via the objective function, i.e., thorough the locations of the best positions found so far *pbest* and *gbest*. In order to understand the behavior of a complex system, it often helps to begin by examining a simpler version of it. Thus, without loss of generality, the algorithm description can be reduced to the one-dimensional case:

$$v_{k+1} = wv_k + c_1 rand(\,)(pbest \quad s_k) + c_2 rand(\,)(gbest \quad s_k)$$
(5)

$$s_{k+1} = s_k + v_{k+1} \qquad (6)$$

For the theoretical analysis of the PSO, the deterministic version will be considered. The deterministic version is obtained by setting the random numbers to their expected values:

$$rand(\,) = rand(\,) = ½ \qquad (7)$$

Thus, Equation (7) can be simplified using the notation:

$$c = \frac{c_1 + c_2}{2}, \qquad (8)$$

$$p = \frac{c_1}{c_1 + c_2} pbest + \frac{c_2}{c_1 + c_2} gbest \qquad (9)$$

Using this notation, the deterministic PSO algorithm can be expressed as:

$$v_{k+1} = wv_k + c(p \quad s_k) \qquad (10)$$

$$s_{k+1} = s_k + v_{k+1} \qquad (11)$$

The algorithm described by Equation (10) and (11) contains two tuning parameters, w and c, that are truly influence for PSO performance.

In order to analyze dynamic system of PSO, Equation (10) and (11) can be combined and written in compact matrix form as follows:

$$z_{k+1} = Az_k + Bp \quad \text{with}$$

$$z_k = \begin{matrix} v_k \\ s_k \end{matrix}, \quad A = \begin{matrix} w & c \\ w & 1 \quad c \end{matrix}, \quad B = \begin{matrix} b \\ b \end{matrix} \qquad (12)$$

In the context of dynamic system theory, z_k is the particle *state* made up of its current position and velocity, A is the *dynamic matrix* whose properties determine the time behavior of the particle, p is the *external input* used to drive the particle towards a specified position and B is the *input matrix* that gives the effect of the external input on the particle state.

Standard results from dynamic system theory say that the time behavior of the particle depends on the eigenvalues of the dynamic matrix A. The eigenvalues λ_1 and λ_2 (either real or complex) are the solutions of characteristic polynomial equation:

$$\lambda^2 + (c - w - 1)\lambda + w = 0 \tag{13}$$

The necessary and sufficient condition to be stable and will converge is that both eigenvalues of the matrix A have magnitude less than 1. The analysis of the roots of Equation (13) leads to the following set conditions:

$$w < 1, \quad c > 0, \quad \text{and } 2w - c + 2 > 0 \tag{14}$$

The convergence domain in the (w,c) plane is the triangle shown in Fig. (2). For any initial position and velocity, the particle will converge to its equilibrium position, p, as in Equation. (10) if and only if the algorithm parameters are selected inside this triangle.

Before convergence, the particle exhibits harmonic oscillation around the equilibrium point when the eigenvalues of the matrix A, which are also the roots of Equation (14), are complex. This equivalent to:

$$w^2 + c^2 - 2wc - 2w - 2c + 1 < 0 \tag{15}$$

The corresponding domain in the (w,c) plane is elliptical function as depicted in Fig. (2).

The particle may also exhibit zigzagging behavior around the equilibrium point when at least one of the eigenvalues of the matrix A, whether real or complex, has negative real part. This is equivalent to:

$$w < 0 \text{ or } w - c + 1 < 0 \tag{16}$$

The corresponding domain in the (w,c) plane is drawn in Fig. (2). Zigzagging may be combined with harmonics oscillation.

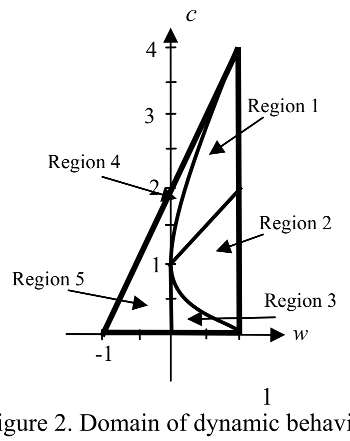

Figure 2. Domain of dynamic behavior in the (w,c) parameter space

2.3. Convergence Analyses

There are several interesting regions for the analyses based on integration of the three figures aforementioned before, such as shown in Fig. (2). In Region 1, the eigenvalues are complex number with real positive number, Re λ_1 and Re $\lambda_2 > 0$. In this region, the particle exhibits harmonic oscillation around the equilibrium point before convergence. In Region 2, the eigenvalues are complex number as well, but with negative real number, Re λ_1 and Re $\lambda_2 < 0$. In this region, the particle exhibits combination of harmonic oscillation and zigzagging around the equilibrium point before convergence. Furthermore, in Region 3, the eigenvalues are positive real number, λ_1 and $\lambda_2 > 0$, where the particle in this region directly convergence to equilibrium point without harmonic oscillation and zigzagging. In Region 4, the eigenvalues are negative real number, λ_1 and $\lambda_2 < 0$. The particle exhibits symmetric zigzagging around the equilibrium point before convergence without harmonic oscillation. Finally, in Region 5, the eigenvalues are positive and negative real number, $\lambda_1 > 0$ and $\lambda_2 < 0$. In this region, the particle exhibits asymmetric zigzagging around the equilibrium point before convergence without harmonic oscillation. Outside of these regions, the particle will diverge. Particle trajectories of these regions are depicted in Fig. (3).

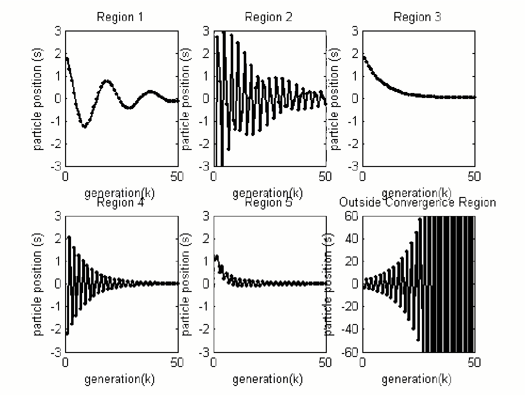

Figure 3. Particle Trajectories and Regions

However, beside determine the shape of particle trajectories, parameter couples agree on speed of convergence. As a general rule, the value of w and c which close to the center of the stability triangle induce quick convergence, while the values close to its borders require many generations to converge, as illustrated in Fig. (4).

According to convergence analysis aforementioned before, a large inertia weight has slow convergence to facilitate a global search, while, a small inertia weight has quick convergence to facilitate a local search. Therefore, by decreasing the inertia weight, w, from a relatively large value to a small one with associated cognitive and social coefficients, c, through the course of the PSO run, the PSO tends to have more global search ability at the beginning of the run and have more local search ability near the end of the run.

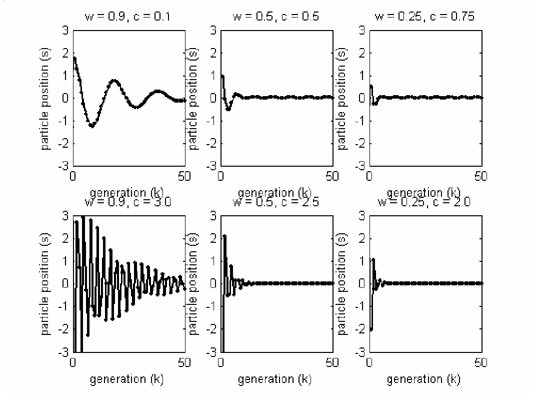

Figure 4. Particle Trajectories and Regions

However, how to decrease inertia weight effectively to achieve good balancing of global and local search is not an easy task. Sigmoid decreasing inertia weight (SDIW) is implemented in this paper to find a better compromise of exploitation-exploration trade-off.

3. PSO with Sigmoid Decreasing Inertia Weight

The present paper proposes a new nonlinear function modulated inertia weight adaptation with time for improved performance of PSO algorithm. Instead of linearly decreasing of inertia weight, the schema attempted to decrease inertia weight by means of sigmoid function, as shown in Fig. 5.

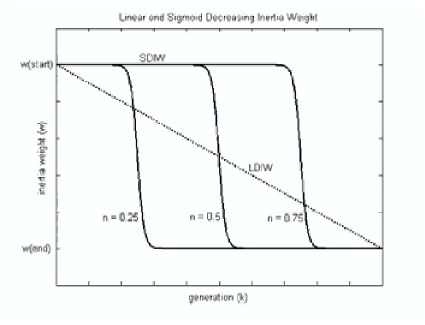

Fig. 5. Sigmoid and Linearly Decreasing Inertia Weight

The proposed function of sigmoid is given as:

$$w_k = \frac{w_{start} \quad (w_{start} \quad w_{end})}{(1+e^{(\quad p*(k \quad n*gen))})} \tag{17}$$

when

$$p = 10^{(\log(gen) \quad 2)} \tag{18}$$

where w_k is inertia weight at k, w_{start} and w_{end} are inertia weight at the start and the final inertia weight at the end of a given run, respectively. Furthermore, p is the constant to adjust sharpness of the function, *gen* is the maximum number of generations to run and n is the constant to set partition of sigmoid function.

In sigmoid, a large inertia weight is maintained in first part of PSO process to assure a global search. Afterwards, a small inertia weight is retained to facilitate a local search in final part of PSO process. There is very short inertia weight gradation between large and small one. This method will provide a balance between global and local searching to give the PSO a superior performance.

In order to further illustrate the effect of this SDIW, some experiments are set. The results are shown and discussed in next sections.

4. Experimental Setting

To illustrate the behavior of the proposed method, four non-linear benchmark functions are used here. The first function is the Sphere function described by Equation (19):

$$f_0(x) = \sum_{i=1}^{n} x_i^2 \tag{19}$$

where $x = [x_1, x_2, ..., x_n]$ is an n-dimensional real-valued vector. The second function is the Rosenbrock function given as:

$$f_1(x) = \sum_{i=1}^{n} (100(x_{i+1} \quad x_i^2)^2 + (x_i \quad 1)^2) \tag{20}$$

The third function is the generalized Rastrigin function formulated as:

$$f_2(x) = \sum_{i=1}^{n} (x_i^2 \quad 10\cos(2\pi x_i) + 10) \tag{21}$$

The fourth function is the generalized Griewank function shown as:

$$f_3(x) = \frac{1}{400}\sum_{i=1}^{n} x_i^2 \quad \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1 \tag{22}$$

For the purpose of evaluation, the asymmetric initialization method is adopted her for the population initialization. Table 1 lists the initialization ranges of the four functions:

Table 1. Asymmetric Initialization Range

Function	Asymmetric Initialization Range
f_0	$(50 , 100)^n$
f_1	$(15 , 30)^n$
f_2	$(2.56 , 5.12)^n$
f_3	$(300 , 600)^n$

For each function, three different dimension sizes are tested. They are dimension sizes: 10, 20 and 30. The maximum number of generations is set as 1000, 1500 and 2000 corresponding to the dimensions 10, 20 and 30, respectively. In order to investigate whether the PSO algorithm scales well or not, different population sizes are used for each function with different dimensions. They are population sizes: 20, 40 and 80. A sigmoid decreasing inertia weight is used which starts at 0.9 and ends at 0.4, which $c_1 = 2$ and $c_2 = 2$. With aim to find the best partition of sigmoid function, different sigmoid constants, n, are used, that are: 0.25, 0.5 and 0.75.

5. Experimental Results

As the main objective of searching methods was to achieve faster speed convergence and better solution accuracy, the experiments results will be shown in table and graphs. Linear Decreasing Inertia Weight (LDIW) used as comparison to proposed method.

Figure 6 shows the results for the sphere function with two different population sizes, respectively. Table 2 lists the mean fitness values of the best particle found for the 30 runs for each function. It is easy to see for the sphere function, that PSO can find the more optima values vary fast and PSO algorithm also scales very well, especially for *n* equal to 0.25.

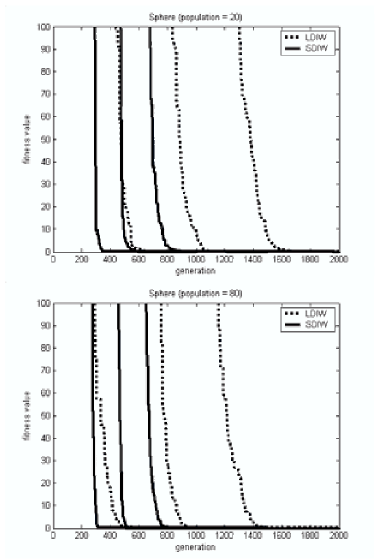

Figure 6. Curve of Sphere function

Table 2. The mean fitness values for the Sphere function

Dim	LDIW	SDIW		
		n = 0.25	**n = 0.5**	**n = 0.75**
Pop = 20				
10	2.6897e-09	5.8519e-35	7.2656e-22	1.3174e-10
20	2.9122e-07	3.5893e-21	9.0725e-08	3.1900e-04
30	5.5721e-05	2.6329e-15	3.7966e-08	3.0000e-03
Pop = 40				
10	4.3735e-18	1.6285e-42	6.8196e-27	1.0791e-11
20	6.2059e-10	1.6278e-28	4.9444e-17	1.3624e-06
30	1.0369e-06	4.6623e-21	1.3846e-12	2.9829e-04
Pop = 80				
10	8.7638e-19	7.3267e-49	1.3887e-31	1.1554e-12
20	5.7939e-12	6.9927e-32	1.8691e-19	1.4061e-09
30	2.1866e-08	5.6160e-25	2.3648e-15	2.6464e-06

Figure 7 shows the results for the Rosenbrock function with two different population sizes, respectively. Figure 8 and Figure 9 show the results for the Rastrigin and Griewank functions with two different population sizes, respectively.

Table 3 to 5 list the mean fitness values of the best particle found for the 30 runs for the other three functions, respectively.

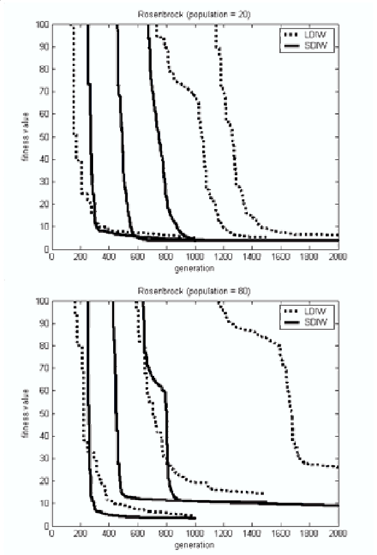

Figure 7. Curve of Rosenbrock function

Table 3. The mean fitness values for the Rosenbrock function

Dim	LDIW	SDIW		
		n = 0.25	**n = 0.5**	**n = 0.75**
Pop = 20				
10	5.0990	3.8001	4.4028	6.2818
20	5.3858	3.9866	4.0420	5.3584
30	6.2131	3.9866	4.0801	5.8690
Pop = 40				
10	4.8889	3.3509	4.0093	5.4973
20	4.3776	4.1118	3.9998	8.6896
30	5.5788	4.4133	4.2580	10.5898
Pop = 80				
10	4.5707	3.3854	3.9582	5.1686
20	14.3539	10.4202	6.5787	14.5359
30	18.3061	8.8637	4.0649	4.1416

By looking at the shape of the curves in all the figures, it easy to see the proposed PSO converges quickly under the all cases and will slow its convergence speed down when reaching the optima values. Nevertheless, the results shown illustrate that by using a SDIW, the performance of PSO can be improved greatly and have better results than LDIW. From the figures, it is also clear that the PSO with different population sizes has almost the similar performance.

6. Conclusion

In this paper, particle trajectories of simple PSO process have been analyzed. Significant parameters, inertia weight and cognitive and/or social constants, have been studied. The sigmoid function to improve PSO performance has been described. The performance of the PSO algorithm with Sigmoid Decreasing Inertia Weight has been investigated and extensively compared with Linear Decreasing Inertia Weight by experimental studies of four nonlinear functions. Experiments results shows that propose method has more robust property with the available PSO method, that is, not sensitive to population size in all four functions test. Moreover, the new PSO method greatly improves the accuracy and convergence speed of search, especially for sigmoid function with *n* equal 0.25.

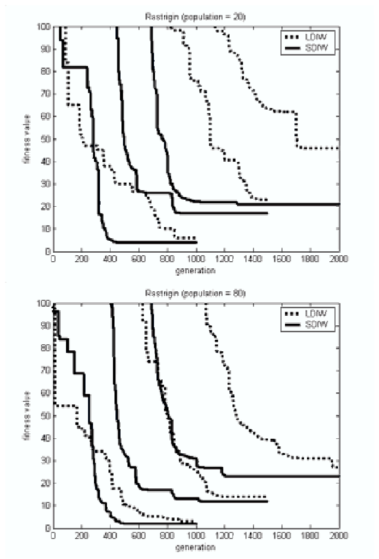

Figure 8. Curve of Rastrigin function

Table 4. The mean fitness values for the Rastrigin function

Dim	LDIW	SDIW		
		n = 0.25	n = 0.5	n = 0.75
Pop = 20				
10	5.9698	3.9798	4.9748	7.9597
20	22.8917	16.9143	19.8992	23.9236
30	45.7682	20.8941	40.7933	49.4858
Pop = 40				
10	3.9798	2.9489	2.9849	4.9760
20	15.9194	11.8387	14.9244	17.2506
30	40.7939	31.8387	38.8033	46.7630
Pop = 80				
10	2.9849	1.9899	3.9798	4.0131
20	13.9294	11.9395	12.9345	15.9195
30	27.4395	22.8840	27.8588	28.9393

Table 5. The mean fitness values for the Griewank function

Dim	LDIW	SDIW		
		n = 0.25	n = 0.5	n = 0.75
Pop = 20				
10	0.0984	0.0836	0.0763	0.0960
20	0.0713	0.0662	0.0636	0.0320
30	0.0742	0.6364	0.0271	0.0999
Pop = 40				
10	0.0640	0.0787	0.0684	0.0935
20	0.0835	0.0711	0.0614	0.1250
30	0.0737	0.0246	0.0197	0.0866
Pop = 80				
10	0.0813	0.0713	0.0615	0.0866
20	0.0492	0.0246	0.0123	0.0710
30	0.0172	0.0148	0.0123	0.0197

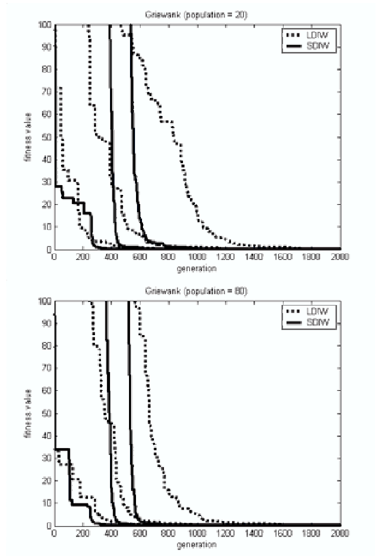

Figure 9. Curve of Griewank function

References

[1] Kennedy, J. and Eberhart, R.C. 1995. Particle swarm optimization, In *Proceeding of IEEE International Conference on Neural Networks*, IV, 1942-1948. Piscataway, NJ: IEEE Service Center.

[2] Eberhart, R. C., and Kennedy, J. 1995. A new optimizer using particle swarm theory, In *Proceeding of the Sixth International Symposium on Micro Machine and Human Science*, 39-43. Nagoya, Japan, Piscataway, NJ: IEEE Service Center.

[3] Shi, Y., Eberhart, R. C. 1998. Parameter selection in particle swarm optimization. In *Evolutionary Programming VII*, 591-600. New York: Springer-Verlag.

[4] Shi, Y., and Eberhart, R.C. 1998. A modified particle swarm optimizer. In *Proceeding of the IEEE International Conference on Evolutionary Computation*, 69-73. NJ: IEEE Press, Piscataway.

[5] Shi, Y., and Eberhart, R. C., 2001. Fuzzy Adaptive Particle Swarm Optimization, In *Proceeding Congress on Evolutionary Computation 2001*, 101-106. Seoul, Korea, Piscataway, NJ: IEEE Service Centre.

[6] Eberhart, R. C., and Shi, Y. 2001. Tracking and optimizing dynamic systems with particle swarms, In *Proceeding Congress on Evolutionary Computation 2001*, Seoul, Korea, Piscataway, NJ: IEEE Service Centre.

[7] Clerc, M., and Kennedy, J., 2002. The Particle Swarm: Explosion, Stability, and Convergence in a Multi-Dimensional Complex Space, *IEEE Transaction on Evolutionary Computation* 6(1):58-73.

[8] Zheng, Y, et. al., 2003. Empirical Study of Particle Swarm Optimizer with an Increasing Inertia Weight, In *Proceeding of the IEEE Congress on Evolutionary Computation 2003,* Vol.1, 221-226.