IJCSES International Journal of Computer Sciences and Engineering Systems, Vol. 5, No. 3, July 2011. pp. 209-217.
Accepted 25 November 2009

Improving Particle Swarm Optimization Convergence with Spread and Momentum Factors

Idris Abd Latiff[1] and M. O. Tokhi[2]

Department of Automatic Control and Systems EngineeringUniversity of Sheffield, Sheffield S1 3JD
United Kingdom, *E-mail:[1]cop06ia@sheffield.ac.uk, [2]o.tokhi@sheffield.ac.uk*

Abstract: *Particle Swarm Optimization (PSO) is a swarm intelligence search method based on the behavior of birds flocking and fish schooling. It is known for its ability to perform fast computation compared to other evolutionary computational methods like Genetic Algorithms. Several parameter control methods have been developed to make the PSO algorithm faster and more accurate such as linearly decreasing inertia weight (LDIW) and time-varying acceleration coefficients (TVAC). This paper presents an improvement over existing techniques by introducing spread factor and momentum factor into the PSO algorithm. Test results show that the PSO with these two factors produce superior performance and suitable for applications where speed and precision are important.*

Key words: *Particle Swarm Optimization, Spread Factor, Momentum Factor.*

1. INTRODUCTION

Particle swarm optimization (PSO) was introduced by James Kennedy and Russell Eberhart in 1995 [1-2] as a population-based search method. In PSO, individuals in the population learn from the member at the best position to reach the group objective. As the population moves towards its objective, each individual will adjust its position according to its own and the neighbor's experiences.

PSO is made up of a population of potential solutions called particles. These particles move in the search space of the objective function towards its optimum point. The main steps in the PSO algorithm are velocity update, position update and fitness evaluation. The ith particle in a d-dimensional search space can be represented by

$$X_i = (x_{i1}, x_{i2}, \ldots, x_{id}) \quad (1)$$

The velocity update equation and position update equation are given respectively as

$$v_{id} = wv_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}) \quad (2)$$

$$x_{id} = x_{id} + v_{id} \quad (3)$$

where v_i is the velocity of particle i, x_i is the position of particle i, w is the inertia weight, c_1 is the cognition factor, c_2 is the social factor, r_1 and r_2 are uniformly distributed random numbers between 0 and 1, p_i is personal best (*pbest*), i.e. the best position of particle i so far, and p_g is global best (*gbest*), i.e. the best particle position in the population so far. The coefficient c is also known as the acceleration factor or learning factor. The second part of Equation (2) is called the cognitive part where particles learn from their own experience, and adjust themselves accordingly. The third part of Equation (2), called the social part, describes how particles interact with their leader to find the objective.

The PSO algorithm can be summarized as follows:

(1) Initialize the position and velocity for each particle.

(2) Evaluate function fitness to determine each particle's personal best and global best particle.

(3) Update personal best if it is better than previous one.

(4) Update global best if it is better than previous one.

(5) Calculate particle velocity using Equation (2).

(6) Update particle position using Equation (3).

(7) Repeat steps (2)-(6) until termination criteria satisfied

2. PARAMETER SELECTION

The behavior of particles in a swarm and its performance are influenced by several factors, which can be classified as follows:

1. Population size or the number of particles in a swarm. More particles mean higher calculation cost but greater chance of finding the global optimum faster. The number of dimensions generally has no effect on performance since each dimension is independent from each other, except in cases where some values are interchanged between dimensions [3].
2. Type of neighborhood, or topology, which can be either global or local.
3. Values, or range of values of inertia weight, constriction factor, acceleration coefficients, and random numbers.
4. Constraints such as velocity limit, maximum number of iterations and desired level of precision,
5. Type of objective function landscape. Some test functions are unimodal with only one global optimum while others are multimodal. Most test functions have local optima.

This paper analyzes the effect of inertia weight and acceleration coefficient on the performance of PSO algorithms. Basically, there are two ways of applying these parameters: fixed and time-varying. Original PSO used the values of w = 1 and c_1 = c_2 = 2 [1]. A constriction factor, κ, which modifies all the terms in the velocity equation was introduced by Clerc [4]. In this case, fixed values of κ = 0.729 and c_1 = c_2 = 2.05 are used according to the following equation:

$$v_{id} = \kappa(v_{id} + c_1 r_1(p_{id} - x_{id}) + c_2 r_2(p_{gd} - x_{id})) \quad (4)$$

Time decreasing inertia weight was suggested by Shi and Eberhart [5] to improve the exploration and exploitation phases of the search. PSO algorithms with linearly decreasing inertia weight (LDIW) using an initial value between 0.7 and 0.9, and a final value between 0.1 and 0.6 have been proposed and tested in many empirical studies [6] [7] [8]. Time-varying acceleration coefficient (TVAC) was suggested by Ratnaweera, Halgamuge, and Watson [9]. Both LDIW and TVAC employ the format of linear equation as described by

$$w_t = (w_1 - w_2)\frac{\max\_t - t}{\max\_t} + w_2 \quad (5)$$

$$c_{1t} = (c_{1f} - c_{1i})\frac{t}{\max\_t} + c_{1i} \quad (6)$$

$$c_{2t} = (c_{2f} - c_{2i})\frac{t}{\max\_t} + c_{2i} \quad (7)$$

where w_t is the current value of the inertia weight, w_1 is the initial value of the inertia weight, w_2 is the final value of the inertia weight, c_t is the current value of the acceleration factors, c_f is the final value of the acceleration factors, c_i is the initial value of the acceleration factors, t is the current iteration value and max_t is the maximum iteration.

3. AN ADAPTIVE APPROACH TO INERTIA WEIGHT MODIFICATION

In this paper, a method of influencing the behavior of particles during exploration and exploitation is proposed. The idea of linearly decreasing inertia weight is that w is supposed to assume a high value during exploration and a small value during exploitation. However, decreasing the value of w linearly is a slow process and does not consider the positions of particles in its calculation. The choice of initial and final values in equations (5), (6) and (7) is based on empirical studies which may be suitable in some applications but less effective in others. Hence, an adaptive method employing a parameter called the spread factor (SF) has been used.

SF measures the distribution of particles in the search space called spread, and the distance between average particle position and the global best position called deviation. These two data are measured at every iteration and used to increase or decrease the inertia weight. The performance of PSO using the spread factor can be further improved by adding another parameter called the momentum factor, m. This factor maintains the search momentum even though the spread and deviation of the swarm is approaching zero. A combination of these two factors is expected to produce a level of performance comparable to existing PSO algorithms. To verify the effectiveness of this approach, PSO algorithms using the spread and momentum factors were tested and compared to popular versions of PSO algorithms.

Tests are performed on five benchmark functions, namely:

1. Sphere:
   $f(x) = x^2$, x = [–100, 100]
2. Rastrigin:
   $f(x) = x^2 - 10\cos(2\pi x) + 10$, x = [–5.12, 5.12]
3. Griewank:
   $$f(x) = \frac{1}{4000}x^2 - \cos(x) + 1 \text{ , } x = [-100, 100]$$
4. Ackley:
   $$f(x) = -20\exp\left(-0.2\sqrt{x^2}\right)$$
   $$-\exp(\cos(2\pi x)) + 20 + e, \; x = [-20, 20]$$
5. Schwefel:
   $$f(x) = 418.9829 - x\sin\left(|x|^{1/2}\right), \; x = [0, 600]$$

The tests are done using 20 particles in 1 dimension with maximum iteration of 1000. The stopping criterion used is epsilon = 10^{-9} i.e. the particles are considered to have successfully converged when the maximum spread and deviation from *gbest* were less than epsilon. The algorithm is run 100 times for each test function, and the results are averaged and presented in respective tables.

3.1. PSO Algorithm with the Spread Factor

The spread factor (SF) was introduced to balance the behavior of particles during exploration and exploitation phases [10]. This factor continuously measures the distribution of particles during the search process and modifies the inertia weight according to the equation

$$w = exp(-iter/(SF \times iteration\_max)) \quad (8)$$

where *iter* and *iteration_max* represent current iteration and maximum number of iterations respectively. Equation (8) is an exponentially decreasing function with a maximum value of 1. The decreasing rate is proportional to the value of SF, which in turn is influenced by the spread and deviation of particles in the search space. Initial applications using PSO with SF have shown some promising results [11-12], but further improvements are necessary to make it more effective.

Four different algorithms were compared. For PSO algorithm with SF, three variants were tested. The first one was with $c_1 = c_2 = 2$. The second one was with c_1 linearly decreasing as described by Equation (5) and $c_2 = 2$. The third one was with both c_1 and c_2 linearly decreasing according to Equation (5). For PSO algorithm with linearly decreasing inertia weight (LDIW), two variants were tested. The first one was with $c_1 = c_2 = 2$ and the second one was with both c_1 and c_2 linearly decreasing as described by Equation (9). The third PSO algorithm used Clerc's constriction factor $\kappa = 0.729$ and $c_1 = c_2 = 2.05$. The fourth PSO algorithm used $w = 0.5 + Rand/2$ and $c_1 = c_2 = 2$. The results are presented in Tables 1-5.

$$c = 2 \times (1-(iter/iteration\_max)) \quad (9)$$

Table 1 shows that for the Sphere function, PSO algorithm with SF and both c_1 and c_2 linearly decreasing was the fastest with 101 iterations. However, PSO algorithm with LDIW and $c_1 = c_2 = 2$ achieved the smallest value of global minimum of 1.97650×10^{-14}. All PSO algorithms achieved 100% convergence except the one with $w = 0.5 + Rand/2$, which achieved only 89%.

Table 2 shows that for the Rastrigin function, PSO algorithm with SF and both c_1 and c_2 linearly decreasing was the fastest with 122 iterations. PSO algorithm using $w = 0.5 + Rand/2$ and $c_1 = c_2 = 2$ was able to obtain the smallest global minimum of 5.34999×10^{-12} but its success rate was only 90% while with others it was 100%.

Table 3 shows that for the Griewank function, PSO algorithm with SF and both c_1 and c_2 linearly decreasing was the fastest with 176 iterations. The smallest global minimum of -1.35105×10^{-10} was found by PSO algorithm with LDIW and both c_1 and c_2 linearly decreasing. All PSO algorithms achieved 100% convergence except the one with $w = 0.5 + Rand/2$, which achieved only 50%.

Table 4 shows that for the Ackley function, PSO algorithm with SF and both c_1 and c_2 linearly decreasing was the fastest with 98 iterations. PSO algorithm with LDIW and both c_1 and c_2 linearly decreasing gave the smallest global minimum of -4.61136×10^{-16}. In this case, the success rate of all PSO algorithm was 100% except with that using $w = 0.5 + Rand/2$ which was 96%.

Table 5 shows the result for the Schwefel function. It is important because it proves that only PSO with SF, linearly decreasing c_1 and constant c_2 was able to maintain 100% success rate under the given test conditions. The algorithm, however, required 522 iterations against 260 needed by PSO with SF and both c_1 and c_2 linearly decreasing.

A few observations can be made from the test results. The most important one is that employing

the spread factor ensures fast convergence in all cases. The effects of linearly decreasing the acceleration coefficient can be seen from both the speed and success rate. By linearly decreasing the cognitive factor only, the number of iterations can be reduced significantly. Tables 1-5 show the reduction of 7.8%, 12.6%, 18.6%, 7.3%, and 0.95% respectively. By linearly decreasing both cognitive and social factors, further reduction of 4.7%, 7.6%, 14.5%, 3.9%, and 50.2% can be achieved. In the case of the Schwefel function (Table 5), linearly decreasing both c_1 and c_2 does not guarantee convergence. As such, the recommended variant of PSO with SF is with c_1 linearly decreasing as per Equation (9) and c_2 is kept constant at 2. This parameter set allows fast convergence and avoids the trap of local optima at the same time.

Overall, the results show that PSO with SF is 3 to 5 times faster than PSO with LDIW, and twice as fast compared to PSO with Clerc's constriction factor, while PSO with random values of inertia weight is the slowest in all cases. This demonstrates the importance of proper parameter selection as arbitrary values cannot guarantee convergence at global optima. In any case, convergence at global optima can hardly be guaranteed in stochastic algorithms, but the approach proposed in this study increases the probability of achieving it.

Table 1
Performance Comparison on Sphere Function

Inertia Weight	*c1*	*c2*	*No of Iterations*	*Global Best*	*Success Rate*
SF	2.0	2.0	115	2.29778×10^{-06}	100%
SF	LD 2.0~0.0	2.0	106	-8.69872×10^{-05}	100%
SF	LD 2.0~0.0	LD 2.0~0.0	**101**	-1.34283×10^{-05}	100%
LDIW	2.0	2.0	573	$\mathbf{1.97650 \times 10^{-14}}$	100%
LDIW	LD 2.0~0.0	LD 2.0~0.0	431	-3.93630×10^{-14}	100%
0.729	2.05	2.05	273	2.81290×10^{-14}	100%
0.5+ *Rand*/2	2.0	2.0	661	8.32025×10^{-14}	89%

Table 2
Performance Comparison on Rastrigin Function

Inertia Weight	*c1*	*c2*	*No of Iterations*	*Global Best*	*Success Rate*
SF	2.0	2.0	151	1.14157×10^{-06}	100%
SF	LD 2.0~0.0	2.0	132	-1.09228×10^{-09}	100%
SF	LD 2.0~0.0	LD 2.0~0.0	**122**	2.45155×10^{-08}	100%
LDIW	2.0	2.0	557	1.57899×10^{-11}	100%
LDIW	LD 2.0~0.0	LD 2.0~0.0	414	-1.12254×10^{-11}	100%
0.729	2.05	2.05	265	1.49099×10^{-11}	100%
0.5 + *Rand*/2	2.0	2.0	623	$\mathbf{5.34999 \times 10^{-12}}$	90%

Table 3
Performance Comparison on Griewank Function

Inertia Weight	*c1*	*c2*	*No of Iterations*	*Global Best*	*Success Rate*
SF	2.0	2.0	253	1.50445×10^{-08}	100%
SF	LD 2.0~0.0	2.0	206	4.54612×10^{-05}	100%
SF	LD 2.0~0.0	LD 2.0~0.0	**176**	3.66301×10^{-05}	100%
LDIW	2.0	2.0	663	-3.62390×10^{-10}	100%
LDIW	LD 2.0~0.0	LD 2.0~0.0	489	$\mathbf{-1.35105 \times 10^{-10}}$	100%
0.729	2.05	2.05	390	1.38939×10^{-10}	100%
0.5 + *Rand*/2	2.0	2.0	795	-1.11569×10^{-10}	50%

Table 4
Performance Comparison on Ackley Function

Inertia Weight	*c1*	*c2*	*No of Iterations*	*Global Best*	*Success Rate*
SF	2.0	2.0	110	-2.82560×10^{-06}	100%
SF	LD 2.0~0.0	2.0	102	1.97982×10^{-07}	100%
SF	LD 2.0~0.0	LD 2.0~0.0	**98**	3.81296×10^{-07}	100%
LDIW	2.0	2.0	548	6.82203×10^{-14}	100%
LDIW	LD 2.0~0.0	LD 2.0~0.0	403	$\mathbf{-4.61136 \times 10^{-16}}$	100%
0.729	2.05	2.05	238	-7.62216×10^{-15}	100%
0.5 + *Rand*/2	2.0	2.0	574	-2.33674×10^{-15}	96%

Table 5
Performance Comparison on Schwefel Function

Inertia Weight	*c1*	*c2*	*No of Iterations*	*Global Best*	*Success Rate*
SF	2.0	2.0	527	420.9687469906	66%
SF	LD 2.0~0.0	2.0	522	420.9687466007	**100%**
SF	LD 2.0~0.0	LD 2.0~0.0	**260**	420.9688283726	55%
LDIW	2.0	2.0	804	420.9687463651	26%
LDIW	LD 2.0~0.0	LD 2.0~0.0	568	420.9687463606	20%
0.729	2.05	2.05	594	420.9687463561	66%
0.5 + *Rand*/2	2.0	2.0	846	420.9687463808	10%

3.2. PSO Algorithm with the Spread and Momentum Factor

Although the PSO with SF enables fast convergence, the algorithm cannot achieve the global best values obtained by other algorithms. As the particles converge towards global optimum, the value of inertia weight is quickly reduced to zero, and the velocity update equation is effectively given by

$$v_{id} = c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}) \quad (10)$$

Since the value of c_1 is reduced by Equation (9), the swarm is mainly affected by the location of *gbest*. The *gbest* particle is itself a *pbest*. As a result, the global best position can hardly be improved without additional term included in the velocity equation. To keep the swarm exploiting the global optimum, the momentum factor, m, has been added to the velocity equation given as [13]

$$v_{id} = c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}) + m \quad (11)$$

With the addition of the momentum factor, the global best particle has more chances of improving its value. Many PSO algorithms with linearly decreasing inertia weight (LDIW) employ the final value between 0.1 and 0.4 for w. As such, PSO algorithm with SF and m were tested using those values. For each value of m, three variants or different sets of acceleration coefficients were applied. The first variant used a constant value of $c_1 = c_2 = 2$. The second variant used a linearly decreasing value of c_1 as per Equation (9), and a constant value of $c_2 = 2$. The third variant used a linearly decreasing value of c_1 and c_2 as per Equation (9). The results are presented in Tables 6-10.

Table 6 shows that for the Sphere function, the smallest global minimum of -2.21778×10^{-15} was achieved by PSO with $m = 0.2$ and $c_1 = c_2 = 2$ in 237 iterations. All variants gave 100% success rate.

Table 7 shows that for the Rastrigin function, the smallest global minimum of 1.88836×10^{-12} was achieved by PSO with $m = 0.4$ and c_1 and c_2 linearly decreasing in 505 iterations. This variant, however, performed at only 94% success rate while all other variants gave 100% success rate.

Table 8 shows that for the Griewank function, the smallest global minimum of -2.73812×10^{-11} was achieved by PSO with $m = 0.2$ and c_1 and c_2 linearly decreasing in 275 iterations. The variants with $m = 0.1$ and $m = 0.2$ gave 100% success rate. For PSO with SF and m = 0.3, the variant with c_1 and c_2 linearly decreasing gave only 97% success rate. For SPO with m = 0.4, only the variant with c_1 linearly decreasing and $c_2 = 2$ achieved 100% success rate.

Table 6
Performance of PSO with SF and *m* on Sphere Function

Inertia Weight	*c1*	*c2*	*No of Iterations*	*Global Best*	*Success Rate*
SF, m=0.1	2.0	2.0	168	-2.24054×10^{-13}	100%
SF, m=0.1	LD 2.0~0.0	2.0	155	-8.81966×10^{-14}	100%
SF, m=0.1	LD 2.0~0.0	LD 2.0~0.0	**142**	-1.17257×10^{-13}	100%
SF, m=0.2	2.0	2.0	237	$\mathbf{-2.21778 \times 10^{-15}}$	100%
SF, m=0.2	LD 2.0~0.0	2.0	201	4.20601×10^{-14}	100%
SF, m=0.2	LD 2.0~0.0	LD 2.0~0.0	194	-2.54793×10^{-14}	100%
SF, m=0.3	2.0	2.0	322	4.32646×10^{-15}	100%
SF, m=0.3	LD 2.0~0.0	2.0	289	-3.61217×10^{-14}	100%
SF, m=0.3	LD 2.0~0.0	LD 2.0~0.0	267	2.68441×10^{-13}	100%
SF, m=0.4	2.0	2.0	479	-3.83559×10^{-15}	100%
SF, m=0.4	LD 2.0~0.0	2.0	431	4.49156×10^{-14}	100%
SF, m=0.4	LD 2.0~0.0	LD 2.0~0.0	426	1.33057×10^{-14}	100%

Table 7
Performance of PSO with SF and *m* on Rastrigin Function

Inertia Weight	*c1*	*c2*	*No of Iterations*	*Global Best*	*Success Rate*
SF, m=0.1	2.0	2.0	198	-4.97260×10^{-11}	100%
SF, m=0.1	LD 2.0~0.0	2.0	179	2.73054×10^{-11}	100%
SF, m=0.1	LD 2.0~0.0	LD 2.0~0.0	**163**	-5.84345×10^{-11}	100%
SF, m=0.2	2.0	2.0	256	1.25788×10^{-11}	100%
SF, m=0.2	LD 2.0~0.0	2.0	231	-3.95348×10^{-11}	100%
SF, m=0.2	LD 2.0~0.0	LD 2.0~0.0	215	-1.01258×10^{-11}	100%
SF, m=0.3	2.0	2.0	365	3.65821×10^{-12}	100%
SF, m=0.3	LD 2.0~0.0	2.0	325	3.46362×10^{-11}	100%
SF, m=0.3	LD 2.0~0.0	LD 2.0~0.0	301	-2.27230×10^{-11}	100%
SF, m=0.4	2.0	2.0	613	-9.44993×10^{-12}	100%
SF, m=0.4	LD 2.0~0.0	2.0	546	5.89153×10^{-12}	100%
SF, m=0.4	LD 2.0~0.0	LD 2.0~0.0	505	$\mathbf{1.88836 \times 10^{-12}}$	94%

Table 8
Performance of PSO with SF and *m* on Griewank Function

Inertia Weight	*c1*	*c2*	*No of Iterations*	*Global Best*	*Success Rate*
SF, m=0.1	2.0	2.0	291	-1.77771×10^{-10}	100%
SF, m=0.1	LD 2.0~0.0	2.0	236	2.12403×10^{-09}	100%
SF, m=0.1	LD 2.0~0.0	LD 2.0~0.0	**225**	2.87126×10^{-10}	100%
SF, m=0.2	2.0	2.0	365	2.05594×10^{-10}	100%
SF, m=0.2	LD 2.0~0.0	2.0	294	-4.01041×10^{-11}	100%
SF, m=0.2	LD 2.0~0.0	LD 2.0~0.0	275	$\mathbf{-2.73812 \times 10^{-11}}$	100%
SF, m=0.3	2.0	2.0	470	7.21318×10^{-11}	100%
SF, m=0.3	LD 2.0~0.0	2.0	372	-2.36606×10^{-10}	100%
SF, m=0.3	LD 2.0~0.0	LD 2.0~0.0	363	3.37012×10^{-10}	97%
SF, m=0.4	2.0	2.0	696	-1.54714×10^{-10}	98%
SF, m=0.4	LD 2.0~0.0	2.0	584	1.86966×10^{-10}	100%
SF, m=0.4	LD 2.0~0.0	LD 2.0~0.0	516	-1.39092×10^{-10}	53%

Table 9 shows that for the Ackley function, the smallest global minimum of -1.07054×10^{-14} was achieved by PSO with $m = 0.3$ and $c_1 = c_2 = 2$ in 318 iterations. All variants gave 100% success rate.

Table 10 shows the result for the Schwefel function and once again proves that PSO with SF, linearly decreasing c_1 and constant c_2 is very effective under the given test conditions. For m values of 0.1, 0.2 and 0.4 the success rate was 100% while for $m = 0.3$, the success rate was 99% able to maintain 100% success rate. All other variants failed to obtain satisfactory results.

An important observation that can be made from the results presented here is that an increase in the value of m form 0.1 onwards increases the number of iterations significantly without greatly improving the quality of global optimum.

Similar pattern in the reduction of the number of iterations by linearly decreasing the acceleration factor as mentioned in Section 3.1 illustrates how Equ. (9) affects the behavior of particles in the swarm.

Further comparison between PSO with SF only and PSO with SF and m is illustrated in Figures 1-5 where the function fitness values of the global best particle are plotted for two values of m, 0.0 and 0.2. Figure 1 shows that for the Sphere function, PSO with SF and m=0.0 converges to fitness value of 1.1394×10^{-27} in 101 iterations while PSO with SF and m=0.2 converges to the fitness value of 1.4128×10^{-34} in 183 iterations.

Table 9
Performance of PSO with SF and *m* on Ackley Function

Inertia Weight	*c1*	*c2*	*No of Iterations*	*Global Best*	*Success Rate*
SF, m=0.1	2.0	2.0	167	-9.15084×10^{-14}	100%
SF, m=0.1	LD 2.0~0.0	2.0	148	8.95291×10^{-14}	100%
SF, m=0.1	LD 2.0~0.0	LD 2.0~0.0	**141**	7.67943×10^{-14}	100%
SF, m=0.2	2.0	2.0	222	4.73380×10^{-14}	100%
SF, m=0.2	LD 2.0~0.0	2.0	200	-2.04945×10^{-13}	100%
SF, m=0.2	LD 2.0~0.0	LD 2.0~0.0	187	-1.03670×10^{-13}	100%
SF, m=0.3	2.0	2.0	318	**-1.07054×10^{-14}**	100%
SF, m=0.3	LD 2.0~0.0	2.0	281	-1.29007×10^{-14}	100%
SF, m=0.3	LD 2.0~0.0	LD 2.0~0.0	274	2.52440×10^{-14}	100%
SF, m=0.4	2.0	2.0	561	-2.95623×10^{-14}	100%
SF, m=0.4	LD 2.0~0.0	2.0	469	1.09656×10^{-13}	100%
SF, m=0.4	LD 2.0~0.0	LD 2.0~0.0	458	-5.74111×10^{-14}	100%

Table 10
Performance of PSO with SF and *m* on Schwefel Function

Inertia Weight	*c1*	*c2*	*No of Iterations*	*Global Best*	*Success Rate*
SF, m=0.1	2.0	2.0	494	420.9687463585	63%
SF, m=0.1	LD 2.0~0.0	2.0	550	420.9687463635	**100%**
SF, m=0.1	LD 2.0~0.0	LD 2.0~0.0	**309**	420.9687463738	42%
SF, m=0.2	2.0	2.0	565	420.9687463486	64%
SF, m=0.2	LD 2.0~0.0	2.0	558	420.9687463560	**100%**
SF, m=0.2	LD 2.0~0.0	LD 2.0~0.0	323	420.9687463543	43%
SF, m=0.3	2.0	2.0	649	420.9687463672	58%
SF, m=0.3	LD 2.0~0.0	2.0	588	420.9687463612	99%
SF, m=0.3	LD 2.0~0.0	LD 2.0~0.0	368	420.9687463691	26%
SF, m=0.4	2.0	2.0	708	420.9687463494	42%
SF, m=0.4	LD 2.0~0.0	2.0	739	420.9687463513	**100%**
SF, m=0.4	LD 2.0~0.0	LD 2.0~0.0	500	420.9687463604	16%

Figure 2 shows that for the Rastrigin function, PSO with SF and m=0.0 converges to fitness value of 0 in 109 iterations while PSO with SF and m=0.2 converges to the fitness value of 0 in 301 iterations.

Figure 3 shows that for the Griewank function, PSO with SF and m=0.0 converges to fitness value of 0 in 137 iterations while PSO with SF and m=0.2 converges to the fitness value of 0 in 269 iterations.

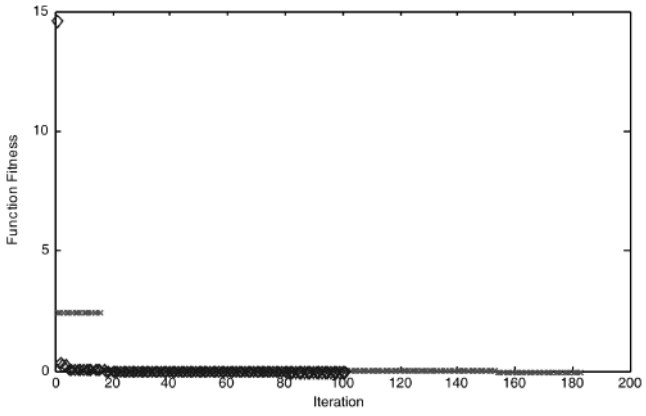

Figure 1: Fitness of *gbest* particle with m=0.0 (diamond) and m = 0.2 (cross) for Sphere Function

Figure 4 shows that for the Ackley function, PSO with SF and m=0.0 converges to fitness value of 6.4668×10^{-12} in 75 iterations while PSO with SF and m=0.2 converges to the fitness value of 7.9048×10^{-14} in 192 iterations.

Figure 5 shows that for the Schwefel function, PSO with SF and m=0.0 converges to fitness value of 1.27276×10^{-05} in 343 iterations while PSO with SF and m=0.2 converges to the fitness value of 1.27276×10^{-05} in 560 iterations.

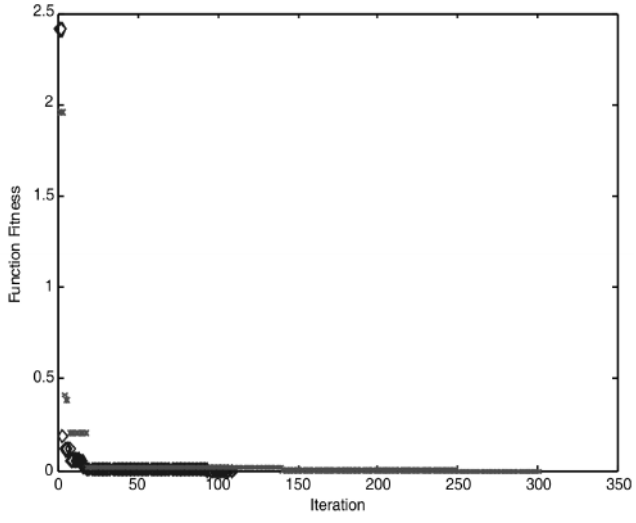

Figure 2: Fitness of *gbest* particle with m=0.0 (diamond) and m = 0.2 (cross) for Rastrigin Function

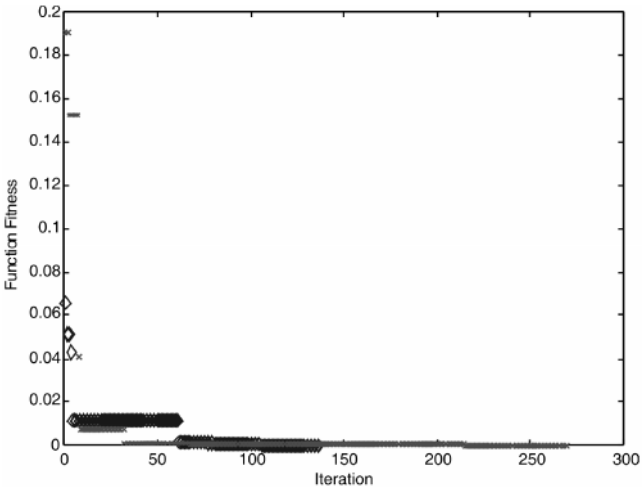

Figure 3: Fitness of *gbest* particle with m=0.0 (diamond) and m = 0.2 (cross) for Griewank Function

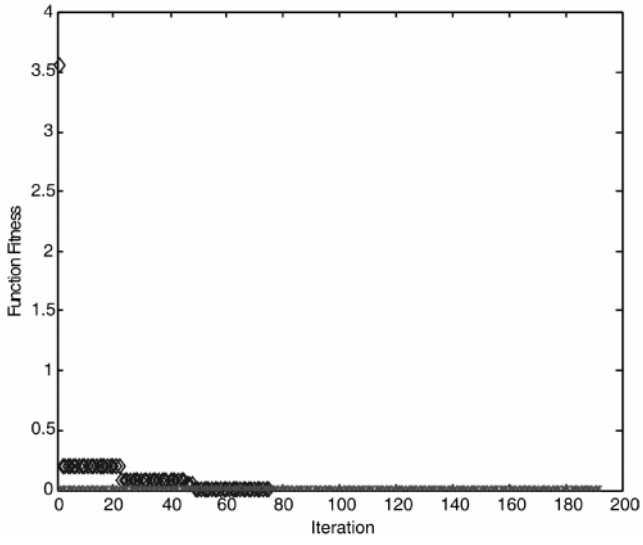

Figure 4: Fitness of *gbest* particle with m=0.0 (diamond) and m = 0.2 (cross) for Ackley Function

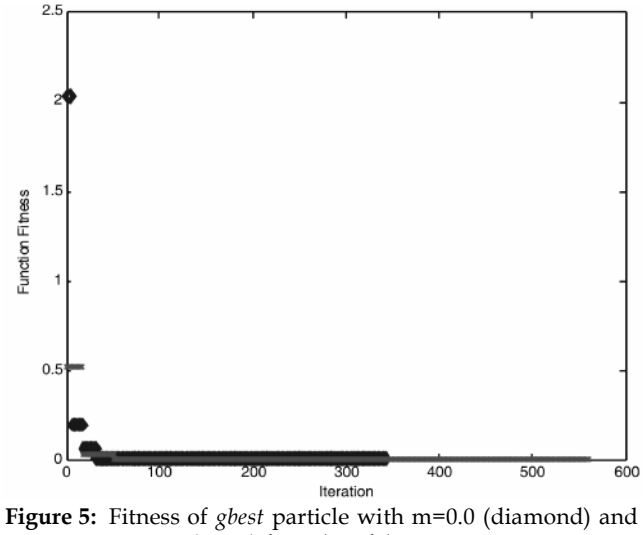

Figure 5: Fitness of *gbest* particle with m=0.0 (diamond) and m = 0.2 (cross) for Schwefel Function

A few observations can be made from the plots. Both algorithms do not require so many iterations during the exploration stage to find the region where the global optima are located. Secondly, the effect momentum factor in delaying convergence can be seen in all plots except in Figure 4. The momentum factor causes the inertia weight to be high for better exploration. It also allows the algorithm to keep fine tuning the fitness value before termination criteria are reached. This results in better fitness in higher numbers of iterations.

4. CONCLUSION

An improvement to the existing PSO algorithms has been proposed by the introduction of the spread and momentum factors. This paper described how these two factors can be used to modify the inertia weight of the velocity equation, and thereby improve the performance of PSO. Tests results have proven that only proper parameter selection can ensure convergence at global optima and consistent performance of the algorithm. However, the experiments done in this study is not comprehensive as it covers only five test functions in 1 dimension. This is due to software limitation. To verify the true effectiveness of the proposed PSO model, more difficult and thorough tests should be conducted.

Alternative methods of keeping the momentum of the global best particle, other than simply assigning a specific momentum factor, should be explored. One consideration is to assign the momentum factor to the *gbest* particle only. Alternatively, the *gbest* particle can be made responsive to the other particles with certain property e.g. having the highest velocity, thus making the algorithm truly adaptive.

REFERENCES

[1] J. Kennedy and R. Eberhart, "Particle swarm optimization", *Proceedings of the IEEE Conference on Neural Networks*, Perth, Australia, 1995, 1942-1948.

[2] R. Eberhart and J. Kennedy, "A New Optimizer using Particle Swarm Theory", *Proceedings of the Sixth International Symposium on Micro Machine and human Science*, Nagoya, Japan, 1995, 39-43.

[3] J. J. Liang, A. K. Qin, P. N. Suganthan and S. Baskar, "Comprehensive Learning Particle Swarm Optimizer for Global Optimization of Multimodal Functions", *IEEE Transactions on Evolutionary Computation*, **10**, 2006, 281-295.

[4] M. Clerc, "The Swarm and the Queen: Towards a Deterministic and Adaptive Particle Swarm optimization", *Proceedings of the Congress on Evolutionary Computation*, Washington DC, USA, 1999, pp. 1951-1957.

[5] R. C. Eberhart and Y. Shi, "Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization", *Proceedings of the Congress on Evolutionary Computation*, La Jolla, USA, 2000, 84-88.

[6] Y. Shi and R. A. Krohling, "Co-evolutionary Particle Swarm Optimization to Solve Min-max Problems", *Proceedings of the Congress on Evolutionary Computation*, Honolulu, USA, 2002, 1682-1687.

[7] R. Brits, A. P. Engelbrecht, and F. van den Bergh, "Locating Multiple Optima using Particle Swarm Optimization", *Applied Mathematics and Computation*, **189**, 2007, 1859-1883.

[8] B. Niu, Y. Zhu and X. He, "Multi-population Cooperative Particle Swarm Optimization", *Lecture Notes in Computer Science*, **3630**, 2005, 874-883.

[9] A. Ratnaweera, S. K. Halgamuge and H. C. Watson, "Self-organizing Hierarchical Particle Swarm Optimizer with Time-varying Acceleration Coefficients", *IEEE Transactions on Evolutionary Computation*, **8**, 2004, 240-255.

[10] I. Abd Latiff and M. O. Tokhi, "Fast Convergence Strategy for Particle Swarm Optimization using spread Factor", *IEEE Conference on Evolutionary Computation*, 2009, 2693-2700.

[11] S. F. Toha, I. Abd Latiff, M. Mohamad and M. O. Tokhi, "Parametric Modeling of a TRMS using Dynamic Spread Factor Particle Swarm Optimization", *Proceedings of 11th International Conference on Computer Modeling and Simulation*, Cambridge, UK, 2009, 95-100.

[12] S. Julai, M.O. Tokhi, M. Mohamad and I. Abd Latiff, "Control of Flexible Plate Structure using Particle Swarm Optimization", *IEEE Conference on Evolutionary Computation*, 2009, 3183-3190.

[13] I. Abd Latiff and M. O. Tokhi, "Improving Particle Swarm Optimization Convergence with Spread and Momentum Factors", *The Second International Conference on Control, Instrumentation and Mechatronic Engineering*, 2009, 406-413.