A New Approach to Improve Particle Swarm Optimization

Liping Zhang, Huanjun Yu, and Shangxu Hu

College of Material and Chemical Engineering, Zhejiang University,
Hangzhou 310027, P.R. China
zhanglp@infotech.zju.edu.cn
yuhj@zjuem.zju.edu.cn
sxhu@mail.hz.zj.cn

Abstract. Particle swarm optimization (PSO) is a new evolutionary computation technique. Although PSO algorithm possesses many attractive properties, the methods of selecting inertia weight need to be further investigated. Under this consideration, the inertia weight employing random number uniformly distributed in [0,1] was introduced to improve the performance of PSO algorithm in this work. Three benchmark functions were used to test the new method. The results were presented to show that the new method is effective.

1 Introduction

Particle swarm optimization (PSO) is an evolutionary computation technique introduced by Kennedy and Eberhart in 1995[1-3]. The underlying motivation for the development of PSO algorithm was social behavior of animals such as bird flocking, fish schooling, and swarm [4]. Initial simulations were modified to incorporate nearest-neighbor velocity matching, eliminate ancillary variable, and acceleration in movement. PSO is similar to genetic algorithm (GA) in that the system is initialized with a population of random solutions. However, in PSO, each individual of the population, called particle, has an adaptable velocity, according to which it moves over the search space. Each particle keeps track of its coordinate in hyperspace, which are associated with the solution (fitness) it has achieved so far. This value is called pbest. Another "best" value is called gbest that is obtained so far by any particle in the population and stored the overall best value.

Suppose that the search space is D-dimensional, then the i-th particle of the swarm can be represented by a D-dimensional vector, $X_i=(x_{i1}, x_{i2},...,x_{iD})$. The velocity of this particle, can be represented by another D-dimensional vector $V_i=(v_{i1}, v_{i2},...,v_{iD})$. The best previously visited position of the i-th particle is denoted as $P_i=(p_{i1}, p_{i2},...,p_{iD})$. Defining g as the index of the best particle in the swarm, then the velocity of particle and its new position will be assigned according to the following two equations:

$$v_{id} = v_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}) \quad (1)$$

E. Cantú-Paz et al. (Eds.): GECCO 2003, LNCS 2723, pp. 134–139, 2003.

$$x_{id} = x_{id} + v_{id} \tag{2}$$

where c_1 and c_2 are positive constant, called acceleration, and r_1 and r_2 are two random numbers, uniformly distributed in [0,1].

Velocities of particles on each dimension are clamped by a maximum velocity V_{max}. If the sum of accelerations would cause the velocity on that dimension to exceed V_{max}, which is a parameter specified by the user, then the velocity on that dimension is limited to V_{max}. V_{max} influences PSO performance sensitively. A larger V_{max} facilitates global exploration, while a smaller V_{max} encourages local exploitation [5].

The PSO algorithm is still far from mature, many authors have modified the original version. Firstly, in order to better control exploration, an inertia weight in the PSO algorithm was first introduced in 1998 [6]. Recently, for insuring convergence, Clerc proposed the use of a constriction factor in the PSO [7]. Equation (3), (4), and (5) describes the modified algorithm.

$$v_{id} = \chi(wv_{id} + c_1 r_1(p_{id} - x_{id}) + c_2 r_2(p_{gd} - x_{id})) \tag{3}$$

$$x_{id} = x_{id} + v_{id} \tag{4}$$

$$\chi = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|} \tag{5}$$

where w is the inertia weight, and χ is a constriction factor, and $\varphi = c_1 + c_2, \varphi > 4$.

The use of the inertia weight for controlling the velocity has resulted in high efficiency for PSO. Suitable selection of the inertia weight provides a balance between global and local explorations. The performance of PSO using an inertia weight was compared with performance using a constriction factor [8], and Eberhart et al. concluded that best approach is to use the constriction factor while limiting the maximum velocity V_{max} to the dynamic range of the variable X_{max} on each dimension. For example, $V_{max} = X_{max}$.

In this work, we proposed a method using random number inertia weight called RNM to improve the performance of PSO.

2 The Ways to Determine the Inertia Weight

As mentioned precedingly, the inertia weight was found to be an important parameter to PSO algorithms. However, the determination of inertia weight is still an unsolved problem. Shi et al. provided methods to determine the inertia weight. In their earlier work, inertia weight was set as constant [6]. By setting maximum velocity to be 2.0, it was found that PSO with an inertia weight in the range [0.9, 1.2] on average has a better performance. In a later work, inertia weight was set to be continuously decreased linearly during run [9]. Still later, a time decreasing inertia weight from 0.9 to 0.4 was found to be better than a fixed inertia weight. The linearly decreasing inertia

weight (LDW) was used by many authors so far [10-12]. Recently another approach was suggested to use a fuzzy variable to adapt the inertia weight [12,13]. The results reported in their papers showed that the performance of PSO can be significantly improved. However, it is relatively complicated.

The right side of equation (1) consists of three parts: the first part is the previous velocity of the particle; the second and third parts are contributing to the change of the velocity of a particle. Shi and Eberhart concluded that the role of the inertia weight w is considered to be crucial for the convergence of PSO [6]. A larger inertia weight facilitates global exploration (searching new areas), while a smaller one tends to facilitate local exploitation. A general rule of thumb suggests that it is better to initially set the inertia weight to a larger value, and gradually decrease it. Unfortunately, the phenomenon that the global search ability is decreasing when inertia weight is decreasing to zero indicates that inertia weight may exit some unclear mechanism [14]. However, the deceased inertia weight is subject to trap the algorithms into the local optima and slows the convergence speed when it is near a minimum. Under this consideration, many cases were tested, and we finally set the inertia weight as random numbers uniformly distributed in [0,1], which is more capable of escaping from the local optima than LDW, therefore better results were obtained. Our motivation is that local exploitation combining with global exploration can be processing parallel. The new version is:

$$v_{id} = r_0 v_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}) \tag{6}$$

where r_0 is a random number uniformly distributed in [0,1], and the other parameters are same as before.

Our method can overcome two drawbacks of LDW. For one thing, decreasing the dependence of inertial weight on the maximum iteration that is difficultly predicted before experiments. Another is avoiding the lacks of local search ability at early of run and global search ability at the end of run.

3 Experimental Studies

In order to test the influence of inertia weight on the PSO performance, three nonlinear benchmark functions reported in literature [15,16] were used since they are well known problems. The first function is the Rosenbrock function:

$$f_1(x) = \sum_{i=1}^{n} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2) \tag{7}$$

where $\boldsymbol{x}=[x_1, x_2,...,x_n]$ is an n-dimensional real-valued vector.

The second is the generalized Rastrigrin function:

$$f_2(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10) \tag{8}$$

The third is the generalized Griewank function:

$$f_3(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1 \quad (9)$$

Three different amounts dimensions were tested: 10, 20 and 30. The maximum numbers of generations were set as 1000, 1500 and 2000 corresponding to the dimensions 10, 20 and 30, respectively. For investigation the scalability of PSO algorithm, three population sizes 20, 40 and 80 were used for each function with respect to different dimensions. Acceleration constants took the values $c_1=c_2=2$. Constriction factor χ =1. For the purpose of comparison, all the V_{max} and X_{max} were assigned by same parameter settings as in literature [13] and listed in table 1. 500 trial runs were taken for each case.

Table 1. X_{max} and V_{max} values used for tests

Function	X_{max}	V_{max}
f_1	100	100
f_2	10	10
f_3	600	600

4 Results and Discussions

Table 2, 3 and 4 listed the mean best fitness value of the best particle found for the Rosenbrock, Rastrigrin, and Griewank function with two inertia weight selecting methods, LDW and RNW respectively.

Table 2. Mean best fitness value for the Rosenbrock function

Population Size	No. of Dimensions	No. of Generations	LDW Method	RNW Method
20	10	1000	106.63370	65.28474
	20	1500	180.17030	147.52372
	30	2000	458.28375	409.23443
40	10	1000	61.36835	41.32016
	20	1500	171.98795	95.48422
	30	2000	289.19094	253.81490
80	10	1000	47.91896	20.77741
	20	1500	104.10301	82.75467
	30	2000	176.87379	156.00258

By comparing the results of two methods, it is clearly to see that the performance of PSO can be improved with random number inertia weight for Rastrigrin and Ro-

senbrock function, while for the Griewank function, results of two methods are comparable.

Table 3. Mean best fitness value for the Rastrigrin function

Population Size	No. of Dimensions	No. of Generations	LDW Method	RNW Method
20	10	1000	5.25230	5.04258
	20	1500	22.92156	20.31109
	30	2000	49.21827	42.58132
40	10	1000	3.56574	3.22549
	20	1500	17.74121	13.84807
	30	2000	38.06483	32.15635
80	10	1000	2.37332	1.85928
	20	1500	13.11258	9.95006
	30	2000	30.19545	25.44122

Table 4. Mean best fitness value for the Griewank function

Population Size	No. of Dimensions	No. of Generations	LDW Method	RNW Method
20	10	1000	0.09620	0.09926
	20	1500	0.03000	0.03678
	30	2000	0.01674	0.02007
40	10	1000	0.08696	0.07937
	20	1500	0.03418	0.03014
	30	2000	0.01681	0.01743
80	10	1000	0.07154	0.06835
	20	1500	0.02834	0.02874
	30	2000	0.01593	0.01718

5 Conclusions

In this work, the performance of the PSO algorithm with random number inertia weight has been extensively investigated by experimental studies of three non-linear functions. Because local exploitation combining with global exploration can be processing parallel, random number inertia weight (RNW) method can obtain better results than linearly decreasing inertia weight (LDW) method. Lacks of local search ability at early stage of run and global search ability at the end of run using linearly decreasing inertia weight method were overcomed. However, only three benchmark problems had been tested. To fully claim the benefits of the random number inertia weight to PSO algorithm, more problems need to be tested.

References

1. J. Kennedy and R. C. Eberhart. Particle swarm optimization. Proc. IEEE Int. Conf. on Neural Networks (1995) 1942–1948
2. R. C. 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. R. C. Eberhart, Simpson, P. K., and Dobbins, R. W. Computational Intelligence PC Tools. Boston, MA: Academic Press Professional (1996)
4. M. M. Millonas. Swarm, phase transition, and collective intelligence. In C.G. Langton, Eds., Artificial life III. Addison Wesley, MA (1994)
5. K. E. Parsopoulos and M. N. Vrahatis. Recent approaches to global optimization problems through particle swarm optimization. Natural Computing 1 (2002) 235–306
6. Y. Shi and R. Eberhart. A modified particle swarm optimizer. IEEE Int. Conf. on Evolutionary Computation (1997) 303–308
7. M. Clerc. The swarm and queen: towards a deterministic and adaptive particle swarm optimization. Proc. Congress on Evolutionary Computation, Washington, DC,. Piscataway, NJ:IEEE Service Center (1999) 1951–1957
8. R. C. Eberhart and Y. Shi. Comparing Inertia weight and constriction factors in particle swarm optimization. In Proc. 2000 Congr. Evolutionary Computation, San Diego, CA (2000) 84–88
9. H. Yoshida, K. Kawata, Y. Fukuyama, and Y. Nakanishi. A particle swarm optimization for reactive power and voltage control considering voltage stability. In G. L. Torres and A. P. Alves da Silva, Eds., Proc. Int. Conf. on Intelligent System Application to Power Systems, Rio de Janeiro, Brazil (1999) 117–121
10. C. O. Ouique, E. C. Biscaia, and J. J. Pinto. The use of particle swarm optimization for dynamical analysis in chemical processes. Computers and Chemical Engineering 26 (2002) 1783–1793
11. Y. Shi and R. Eberhart. Parameter selection in particle swarm optimization. Proc. 7th Annual Conf. on Evolutionary Programming (1998) 591–600
12. Y. Shi, and Eberhart, R. Experimental study of particle swarm optimization. Proc. SCI2000 Conference, Orlando, FL (2000)
13. Y. Shi and R. Eberhart. Fuzzy adaptive particle swarm optimization. 2001. Proceedings of the 2001 Congress on Evolutionary Computation, vol. 1 (2001) 101–106
14. X. Xie, W. Zhang, and Z. Yang. A dissipative particle swarm optimization. Proceedings of the 2002 Congress on Evolutionary Computation, Volume: 2 (2002) 1456–1461
15. J. Kennedy. The particle swarm: social adaptation of knowledge. Proc. IEEE International Conference on Evolutionary Computation (Indianapolis, Indiana), IEEE Service Center, Piscataway, NJ (1997) 303–308
16. P. J. Angeline. Using selection to improve particle swarm optimization. IEEE International Conference on Evolutionary Computation, Anchor age, Alaska, May (1998) 4–9
17. J. Kennedy, R.C. Eberhart, and Y. Shi. Swarm Intelligence, San Francisco: Morgan Kaufmann Publishers (2001)