A New Particle Swarm Optimization Technique

Chunming Yang and Dan Simon
Electrical and Computer Engineering Department
Cleveland State University
Cleveland, Ohio 44115
c.yang@csuohio.edu

Abstract

In this paper, a new particle swarm optimization method (NPSO) is proposed. It is compared with the regular particle swarm optimizer (PSO) invented by Kennedy and Eberhart in 1995 based on four different benchmark functions. PSO is motivated by the social behavior of organisms, such as bird flocking and fish schooling. Each particle studies its own previous best solution to the optimization problem, and its group's previous best, and then adjusts its position (solution) accordingly. The optimal value will be found by repeating this process. In the NPSO proposed here, each particle adjusts its position according to its own previous worst solution and its group's previous worst to find the optimal value. The strategy here is to avoid a particle's previous worst solution and its group's previous worst based on similar formulae of the regular PSO. Under all test cases, simulation shows that the NPSO always finds better solutions than PSO.

1. Introduction

Particle swarm optimization has been used to solve many optimization problems since it was proposed by Kennedy and Eberhart in 1995 [4]. After that, they published one book [9] and several papers on this topic [5][7][13][15], one of which did a study on its performance using four nonlinear functions adopted as a benchmark by many researchers in this area [14]. In PSO, each particle moves in the search space with a velocity according to its own previous best solution and its group's previous best solution. The dimension of the search space can be any positive integer. Following Eberhart and Kennedy's naming conventions, *D* is the dimension of the search space. The i^{th} particle is represented as $X_i = (x_{i1}, x_{i2}, ..., x_{iD})$ and its previous best is $P_i = (p_{i1}, p_{i2}, ..., p_{iD})$. The index of the best in the whole group is g and the position change is represented as $\Delta X_i = (\Delta x_{i1}, \Delta x_{i2}, ..., \Delta x_{iD})$. Each particle updates its position with the following two equations:

$$\Delta x_{id} = \Delta x_{id} + c_1 rand1(\ )(p_{id} - x_{id}) + c_2 rand2(\ )(p_{gd} - x_{id}) \qquad (1.1)$$

$$x_{id} = x_{id} + \Delta x_{id} \qquad (1.2)$$

where c_1 and c_2 are positive constants, $rand1(\ )$ and $rand2(\ )$ are random numbers between 0 and 1.

Some researchers have found out that setting c_1 and c_2 equal to 2 gets the best overall performance.

The original PSO described above is basically developed for continuous optimization problem. However, lots of practical engineering problems are formulated as combinational optimization problem. Kennedy and Eberhart developed a discrete binary version of PSO for these problems [8]. They proposed a model wherein the probability of a particle's deciding yes or no, true or false, or making some other decision, is a function of personal and social factors as follows:

$$P(x_{id}(t)=1) = f(x_{id}(t-1), v_{id}(t-1), p_{id}, p_{gd}) \qquad (1.3)$$

where

- $P(x_{id}(t)=1)$ is the probability that individual i will choose 1 for the bit at the d^{th} site on the bit string
- $x_{id}(t)$ is the current state of the bit string site d of individual i.
- t means the current time step, and $t-1$ is the previous step
- $v_{id}(t-1)$ is a measure of the individual's predisposition or current probability of deciding 1
- p_{id} is the best state found so far; for example, it is 1 if the individual's best success occurred when x_{id} was 1 and 0 if it was 0
- p_{gd} is the neighborhood best, again 1 if the best success attained by any member of the neighborhood was when it was in the 1 state and 0 otherwise

Both continuous and binary PSO have been used successfully in many optimization problems [1][2][3][10][12], including the famous traveling salesperson problem [11][16].

In our personal experience we know that an individual not only learns from his or her own and other individuals' previous best, but also learns from his or her own and other individuals' mistakes. The idea of NPSO is based on this social behavior. Each particle tries to leave its previous worst position and its group's previous worst position. $P_i = (p_{i1}, p_{i2}, ..., p_{iD})$ represents a particle's previous worst, and the index of the worst in the whole group is g, and the position change is represented as $\Delta X_i = (\Delta x_{i1}, \Delta x_{i2}, ..., \Delta x_{iD})$ .The equations adopted here are similar to those of PSO

$$\Delta x_{id} = \Delta x_{id} + c_1 rand1()(x_{id} - p_{id}) + c_2 rand2()(x_{id} - p_{gd}) \qquad (1.4)$$

$$x_{id} = x_{id} + \Delta x_{id} \qquad (1.5)$$

$c_1, c_2, rand1(\ )$ and $rand2(\ )$ have the same meaning as in PSO.

Binary NPSO is similar to its binary PSO counterpart. The key difference is the reference particle and the records to remember.

2. Comparison Functions

Comparison functions adopted here are four benchmark functions used by many researchers. They are the sphere, Griewank, Rastrigrin and Rosebrock functions. The definition of the sphere function is

$$f(x_i) = \sum_{i=1}^{n} x_i^2 \tag{2.1}$$

where n is the dimension of the sphere. The n-dimensional Griewank function is defined as

$$f(x_i) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\frac{x_i}{\sqrt{i}} + 1 \tag{2.2}$$

The definition of the Rastrigrin function is

$$f(x_i) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10) \tag{2.3}$$

The definition of the Rosenbrock function is

$$f(x_i) = \sum_{i=1}^{n} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2) \tag{2.4}$$

To make a fair comparison, equations of both PSO and NPSO set c_1 and c_2 to two. The simulation runs these two sets of equations at the same time so that the initial swarms are generated only once and given to both PSO and NPSO. The function of $rand1(\ )$ is called once, and the $rand2(\ )$ is called once, for both PSO and NPSO to give a fair comparison. Three different dimensions of each function are used for comparison: two, five and 10. The range of the particles is set to be [-50 50] so that the search time is not too long. The changes of positions are limited to avoid missing optimal values by taking big steps; however, this might be beneficial for jumping out of local optima.

3. Simulation Results

Tables 1-3 show the preliminary results of the comparisons. Each test is run 10 times and each run loops 100 generations

Table 1: Best mean cost in search dimension of two.

Population	Sphere PSO/*NPSO*	Griewank PSO/*NPSO*	Rastringrin PSO/*NPSO*	Rosenbrock PSO/*NPSO*
100	261.1529 *56.0209*	1.2375 *0.0838*	404.9046 *69.8872*	65.5275 *22.6204*
1000	88.5258 *12.6660*	0.8444 *0.0305*	133.7713 *27.3743*	76.8000 *4.2190*
10000	64.5865 *6.5582*	0.6706 *0.0192*	97.5040 *8.6454*	76.5074 *0.8340*

Table 2: Best mean cost in search dimension of five.

Population	Sphere PSO/*NPSO*	Griewank PSO/*NPSO*	Rastringrin PSO/*NPSO*	Rosenbrock PSO/*NPSO*
100	6321.4 *1139.5*	3.5225 *1.0352*	6459.6 *1399.3*	2.0263e+8 *2.0751e+7*
1000	2889.5 *311.6544*	3.0008 *0.5421*	2947.1 *458.4224*	3.2813e+7 *1.8068e+6*
10000	1327.6 *171.0362*	2.5329 *0.4908*	1504.8 *244.9610*	5.1675e+6 *2.2850e+5*
100000	716.5988 *78.6379*	2.0856 *0.3119*	796.8383 *117.0556*	1.9725e+6 *4.4368e+4*

Table 3: Best mean cost in search dimension of ten.

Population	Sphere PSO/*NPSO*	Griewank PSO/*NPSO*	Rastringrin PSO/*NPSO*	Rosenbrock PSO/*NPSO*
100	2.2585e+4 *8.1412e+3*	6.8023 *3.7419*	2.2827e+4 *1.3020e+4*	3.7694e+9 *1.7664e+9*
1000	1.5926e+4 *3.7709e+3*	4.9188 *1.9202*	1.6359e+4 *3.0714e+3*	1.8517e+9 *2.1004e+8*
10000	1.2552e+4 *1.8825e+3*	3.8051 *1.8783*	1.5432e+4 *1.8337e+3*	3.9948e+8 *1.5886e+7*
100000	1.0363e+4 *1.1244e+3*	3.2137 *1.2237*	7.7069e+3 *887.1268*	2.8194e+8 *1.9621e+7*

With these particular settings, NPSO finds better solutions than PSO. More seeds are needed when the search dimension gets larger and other parameters might need to change too. For instance, we might need to run more generations since the solutions might get out of local minima with more random numbers generated. Position change limits may be tuned too: we use a lower limit for searching small dimensions since missing the optimum is more important than searching speed but we use a higher limit for searching large dimensions since exploring large area of the search space is more important. Figures 1–8 show typical convergence results of PSO and NPSO. In each case it is seen that NPSO performs better than PSO.

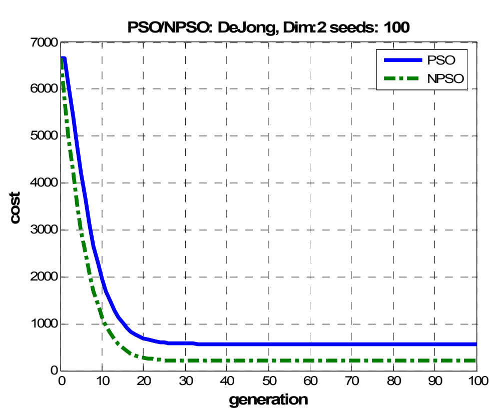

Figure 1

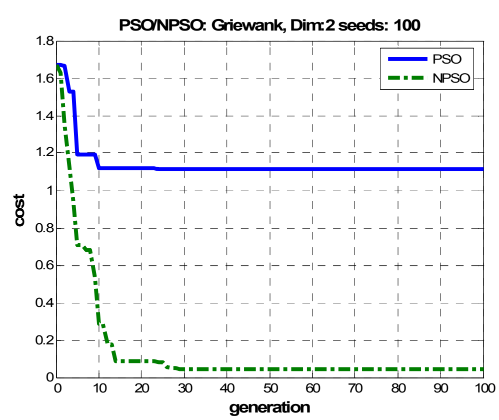

Figure 2

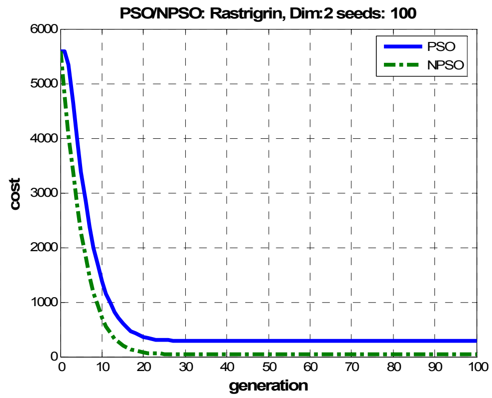

Figure 3

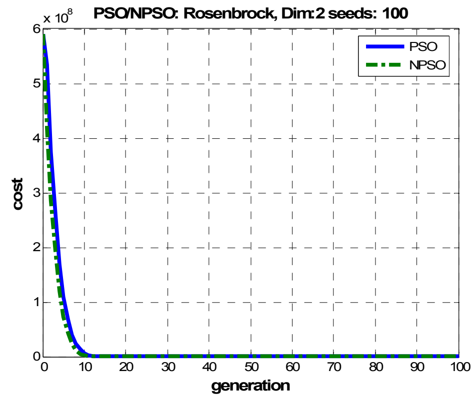

Figure 4

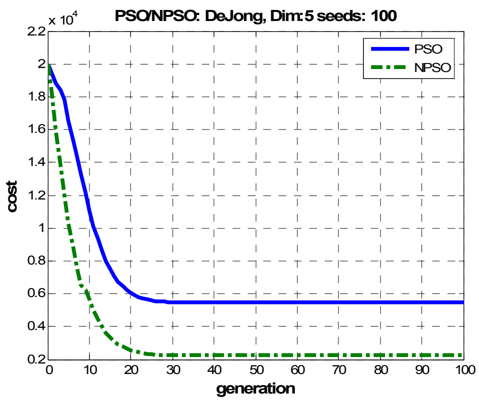

Figure 5

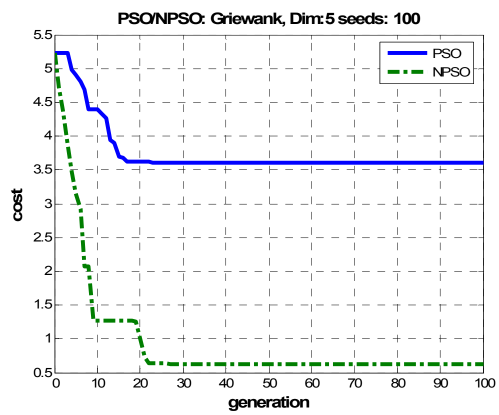

Figure 6

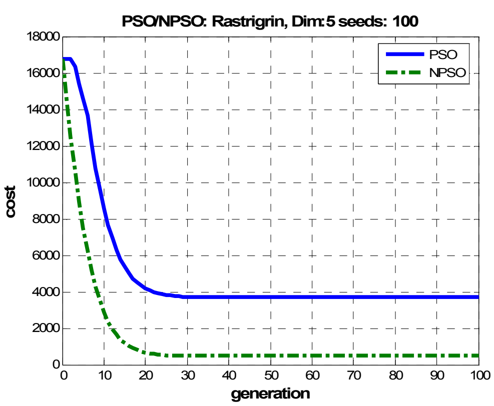

Figure 7

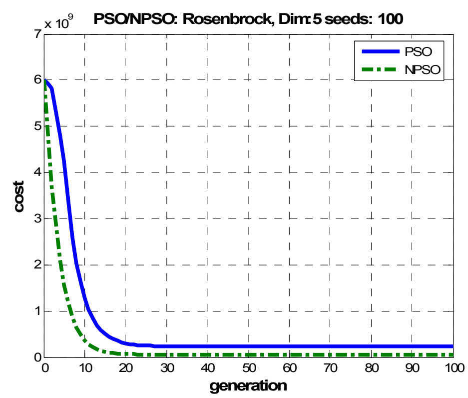

Figure 8

4. Conclusion and Future Research

In some cases, NPSO can find a better solution than PSO. The experiments are limited to a rather narrow setting, so the conclusion here is not comprehensive or definite. NPSO is a variation of regular PSO instead of a better PSO. More research will be done on the conditions under which NPSO does a better job and also on the conditions under which PSO does a better job. Additional work could explore how to combine PSO and NPSO to get an even better search method.

Under the present formulation of PSO and NPSO, each particle moves to a new position regardless of whether the new solution is better than the current one or not. Changes can be made so that it moves to a better solution unconditionally, but moves to a worse position according to some probability. Future research will cover this approach in more detail and try to find some theories as to why this might lead to a better solution.

5. References

[1] Bergh, F. and Engelbrecht, A, (2002) A New Locally Convergent Particle Swarm Optimiser. Conference on Systems, Man and Cybernetics. pp. 96-101

[2] Carlisle, A. and Dozier, G. (2000) Adaptive Particle Swarm Optimization to Dynamic Environment. Proc of International Conference n Artificial Intelligence. pp.429-434

[3] Clarc, M and Kennedy, J (2002). The particle swarm – explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation. pp. 58-73

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

[5] Eberhart, R. C. and Shi, Y. (1998)(b). Comparison between genetic algorithms and particle swarm optimization. In V. W. Porto, N. Saravanan, D. Waagen, and A. E. Eiben, Eds. Evolutionary Programming VII: Proc. 7th Ann. Conf. on Evolutionary Programming Conf., San Diego, CA. Berlin: Springer-Verlag.

[6] Goldberg, D "Genetic Algorithms in search, optimization and machine learning," Addison-Wesley Publishing Company Inc, 1989.

[7] Kennedy, J. and Eberhart, R.C. (1995). Particle swarm optimization. Proc. IEEE Int'l. Conf. on Neural Networks, IV, 1942-1948. Piscataway, NJ: IEEE Service Center.

[8] Kennedy, J. and Eberhart, R.C. (1997). A discrete binary version of the particle swarm algorithm. Proc. 1997 Conf. on Systems, Manand Cybernetics,4104-4109. Piscataway,NJ: IEEE Service Center.

[9] Kennedy, J., Eberhart, R.C., and Shi, Y.(2001). Swarm Intelligence, San Francisco: Morgan Kaufmann Publishers.

[10] Kenedy, J. and Spears, W.M. (1998). Matching algorithms to problems: an experimental test of the particle swam and some genetic algorithms on the multimodal problem generator. Proc. Intl. Conf. on Evolutionary Computation, 78-83. Piscataway, NJ: IEEE Service Center.

[11] Pang. W. Wang, K. Zhou, C., Dong, L. (2004). Fuzzy discrete particle optimization for solving traveling salesman problem. International Conference on Computer and Information Technology. pp. 796-800

[12] Robison, J and Rahmat-Samii, Y (2004). Particle swarm optimization in electromagnetics. Transactions on Antennas and Propagation. Vol. 52, no.2, pp. 397-407

[13] Shi, Y. and Eberhart, R. C. (1998b). A modified particle swarm optimizer. Proceedings of the IEEE International Conference on Evolutionary Computation, 69-73. Piscataway, NJ: IEEE Press.

[14] Shi, Y. and Eberhart, R. C. (1999). Empirical study of particle swarm optimization. Proceedings of the 1999 Congress on Evolutionary Computation, 1945–1950. Piscataway, NJ: IEEE Service Center.

[15] Shi, Y. and Eberhart, R., (2001b). Particle Swarm Optimization with Fuzzy Adaptive Inertia Weight, Proceedings of the Workshop on Particle Swarm Optimization. Indianapolis, IN: Purdue School of Engineering and Technology, IUPUI (in press).

[16] Wang, K. Huang, L. Zhou, C. Pang, W. (2003). Particle swarm optimization for traveling salesman problem. Proceedings of the Second International Conference on Machine Learning and Cybermetics. pp. 1583-1585

[17] PSO Tutorial, http://www.swarmintelligence.org/tutorials.php