IEMS Vol. 6, No. 1, pp. 64-71, June 2007.

An Improvement of Particle Swarm Optimization with A Neighborhood Search Algorithm

Fumihiko Yano†
J. F. Oberlin University 3758 Tokiwa-Machi, Machida, Tokyo 194-0294, Japan
Tel: +81-42-797-2661, Fax: +81-42-797-1887, E-mail: yano@obirin.ac.jp

Tsutomu Shohdohji
Nippon Institute of Technology 4-1 Gakuendai, Miyashiro-Machi, Saitama 345-8501, Japan
Tel: +81-480-33-7717, Fax: +81-480-33-7745 E-mail: shodoji@nit.ac.jp

Yoshiaki Toyoda
Aoyama Gakuin Unversity 5-10-1 Fuchinobe, Sagamihara, Kanagawa 229-8558, Japan
Tel: +81-42-759-6000, Fax: +81-42-759-6560 E-mail: toyoda@ise.aoyama.ac.jp

Received Date, September 2006; Accepted Date, January 2007

Abstract. J. Kennedy and R. Eberhart first introduced the concept called as Particle Swarm Optimization (PSO). They applied it to optimize continuous nonlinear functions and demonstrated the effectiveness of the algorithm. Since then a considerable number of researchers have attempted to apply this concept to a variety of optimization problems and obtained reasonable results. In PSO, individuals communicate and exchange simple information with each other. The information among individuals is communicated in the swarm and the information between individuals and their swarm is also shared. Finally, the swarm approaches the optimal behavior. It is reported that reasonable approximate solutions of various types of test functions are obtained by employing PSO. However, if more precise solutions are required, additional algorithms and/or hybrid algorithms would be necessary. For example, the heading vector of the swarm can be slightly adjusted under some conditions. In this paper, we propose a hybrid algorithm to obtain more precise solutions. In the algorithm, when a better solution in the swarm is found, the neighborhood of a certain distance from the solution is searched. Then, the algorithm returns to the original PSO search. By this hybrid method, we can obtain considerably better solutions in less iterations than by the standard PSO method.

Keywords: Swarm Intelligence, Search Algorithm, Optimization, Meta-hueristics.

1. INTRODUCTION

In the last few decades, meta-heuristics on the basis of life evolution and swarm intelligence have been widely researched. They have been applied to various optimization problems and effective solutions were obtained for certain problems. For example, genetic algorithm (GA), evolutionary programming (EP), and simulated annealing (SA) are applied instead of strict optimization methods for the optimization of complex systems. Recently, as a meta-heuristic method, swarm intelligence has been applied to various optimization problems. Even though each individual in a swarm has low intelligence, the swarm often exhibits high performance with respect to its ability as well as intensive united behavior. This concept of particle swarm optimization (PSO) was first developed and applied to an optimization problem by J. Kennedy and R. Eberhart (Kennedy *et al.*, 1995). PSO is derived primarily from the two main component methodologies; artificial life in swarm, and swarming theory in particular. Since then, many researchers have attemped to apply this concept to a variety of optimization problems and they have obtained reasonable results.

In PSO, individuals communicate and exchange simple information with each other. The information among individuals is communicated in the swarm and the information between individuals and their swarm is also shared. Finally, the swarm approaches to the optimal behavior. PSO utilizes the properties that the individuals exhibit simple behaviors with simple information and the swarm, as an aggregate of individuals, exhibit optimal movement for integrated highly intelli-

† : Corresponding Author

gent information.

Many scientists have attempted to simulate bird flocking and fish schooling behaviors on computers for a long time. C. W. Reynolds was particularly interested in the beauty of bird flocking behavior (Reynolds, 1987). F. Heppner and U. Grenander attempted to determine the rules of this behavior of bird flocking (Heppner *et al*., 1990). They simulated it by maintaining an optimal distance between a bird and its neighbors and obtained very good results. It is possible that this rule can be applied to animal social behavior. E. O. Wilson stated in the article, "In theory at least, individual members of the school can profit from the discoveries and previous experience of all other members of the school during the search for food. This advantage can become decisive, outweighing the disadvantage of competition for food items, whenever the resource is unpredictably distributed in patches" (Wilson, 1975). In other words, social information between members offers an evolutionary advantage. This hypothesis forms the basis of PSO. In 1995, J. Kennedy and R. C. Eberhart proposed PSO using the ideas of bird flocking behavior described above (Kennedy *et al*., 1995).

Further, some researchers compared PSO with genetic algorithms and reported good results. Since PSO has not gained popularity, its properties have not yet been fully elucidated. In this paper, we propose a hybrid algorithm to find more precise solutions in less iterations. In this algorithm, when a better solution for the swarm is obtained, the neighborhood of a certain distance from the solution is searched. The algorithm then returns to the original PSO search. By this hybrid method, we can obtain considerably better solutions than by the standard PSO in less iterations.

2. EVOLUTION OF PSO

PSO is slightly different from existing meta-heuristic algorithms. It was developed on the basis of social behaviors of animals and plants, and physical phenomena; however, it is associated with evolutionary computation as well as associates with GAs and EP.

In this section, the original PSO method is outlined, which was proposed by Kennedy and Everhart (Kennedy *et al*., 1995). The algorithm evolutionarily approaches to the optima in a step by step manner.

First, we define symbols and notations.

- $P_{i,j}$ present position (j-th iteration) of member i in the swarm,
- $\boldsymbol{P}_{i,j}$ vector from the origin to position $P_{i,j}$,
- $V_{i,j}$ objective value at the present position $P_{i,j}$,
- Q_i position that yields the best objective value determined by member i until the j-th iteration,
- $\boldsymbol{Q}_i$ vector from the origin to position Q_i,
- U_i objective value at Q_i (best objective value of member i),
- B position that yields the best objective value determined by whole members of the swarm until the j-th iteration,
- $\boldsymbol{B}$ vector from the origin to position B,
- F objective value at B (best objective value of the swarm),
- $\boldsymbol{T}_{i,j}$ transfer vector of member i at the j-th iteration, and
- c_h random number distributed uniformly between 0 and 1 (distance weight where h = 1, 2).

At the beginning of the algorithm, each member in the swarm is placed randomly in the search domain. The transfer vector of member i at j+1-th iteration, $\boldsymbol{T}_{i,j+1}$, is calculated as follows (refer to Figure 1):

$$\boldsymbol{T}_{i,j+1} = \boldsymbol{T}_{i,j} + 2c_1(\boldsymbol{Q}_i - \boldsymbol{P}_{i,j}) + 2c_2(\boldsymbol{B} - \boldsymbol{P}_{i,j}) \quad (1)$$

Then,

$$\boldsymbol{P}_{i,j+1} = \boldsymbol{P}_{i,j} + \boldsymbol{T}_{i,j+1}$$

If $V_{i,j+1}$ is better than U_i, $U_i \leftarrow V_{i,j+1}$, $\boldsymbol{Q}_i \leftarrow \boldsymbol{P}_{i,j+1}$.

If U_i is better than F, $F \leftarrow U_i$, $\boldsymbol{B} \leftarrow \boldsymbol{Q}_i$.

In the article, several alternative transfer vectors were defined and tested on benchmark problems: it was reported that transfer vector (1) yielded the best results. Later, many researchers showed that equation (1) yielded reasonable results (Kennedy *et al*., 1995).

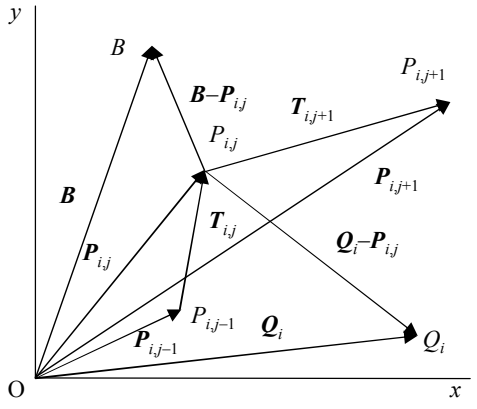

Figure 1. Positions and vectors on a two dimensional X–Y plane

3. MOOT POINTS OF ORIGINAL PSO AND IMPROVEMENTS

The following modified transfer vector has frequently been used in recent researches instead of (1), because it usually yields better solutions (Kennedy *et al*., 2001).

$$\boldsymbol{T}_{i,j+1} = w\boldsymbol{T}_{i,j} + 2c_1(\boldsymbol{Q}_i - \boldsymbol{P}_{i,j}) + 2c_2(\boldsymbol{B} - \boldsymbol{P}_{i,j})$$
$$\text{If } |\boldsymbol{T}_{i,j+1}| > |\boldsymbol{v}| \text{ then } |\boldsymbol{T}_{i,j+1}| \leftarrow |\boldsymbol{v}| \quad (2)$$

where
$\boldsymbol{v}$: maximum transfer vector
w : decrement weight

Occasionally, $P_{i,j+1}$ moves out of the search domain, D. This case was not considered in the original PSO. Although the optimal solution is unknown, it is desirable that the pulling back point is placed near the optimal solution. $P_{i,j+1}$ should be inside D. Therefore, if $P_{i,j+1}$ moves outside D, we must define an operation in PSO to pull back $P_{i,j+1}$ into D.

We have already introduced three types of the pulling back methods (Toyoda *et al.*, 2004; Yano *et al.*, 2004).

1) $P_{i,j+1}$ is pulled back to the nearest point from the present $P_{i,j+1}$ in D,
2) $P_{i,j+1}$ is pulled back to the nearest point where it moves outside of D.
3) $P_{i,j+1}$ is placed randomly in D.

Generally, the position of the optimal solution is unknown. If these methods are dynamically mixed with PSO, we can certainly obtain a more effective method even though the optimal solution can exist at any point. In this paper, we introduce the following approach to PSO.

If $P_{i,j+1}$ moves outside of the search domain D, $P_{i,j+1}$ is replaced as follows:

a) generally, $P_{i,j+1}$ is replaced randomly in D.
b) sometimes, $P_{i,j+1}$ is pulled back to the nearest point where it moves outside of D.

These two steps are switched by a certain probability.

The transfer vector $\boldsymbol{T}_{i,j+1}$ is recalculated by the new $\boldsymbol{P}_{i,j+1}$ as follows:

$$\boldsymbol{T}_{i,j+1} = \boldsymbol{P}_{i,j+1} - \boldsymbol{P}_{i,j}$$

4. NEIGHBORHOOD SEARCH ROUTINE

The length of the transfer vector affects accuracy of the solutions and the computation time. If it is large, broad area within the search domain can be searched in a short computation time; however, a detailed search cannot be carried out. On the contrary, if it is small, it is possible to search in a small area; however, a longer computation time is required to search the entire search domain.

It is very reasonable to assume that when a good solution is found, it is quite within the bounds of possibility that there exist better solutions in its neighborhood.

In this section, we propose a neighborhood search routine and incorporate it in PSO. First, symbols are defined as follows:

$P'_{i,j,e}$ search point at the e-th iteration in the neighborhood search routine, where e = 0, 1, 2, …
$P''_{i,j,e}$ search point that yields the best solution among all the search points at the e-th iteration, where $P''_{i,j,0} = P_{i,j}$,
$\boldsymbol{T}'_{i,j,e+1}$ transfer vector from $P''_{i,j,e}$ to $P'_{i,j,e+1}$, where $\boldsymbol{T}'_{i,j,e+1} = ((-1)^{r1}d, (-1)^{r2}d, \ldots, (-1)^{rn}d)$, r_k = 0 or 1, and k = 1, 2, …, n),
$\boldsymbol{T}''_{i,j,e+1}$ transfer vector from $P''_{i,j,e}$ to $P''_{i,j,e+1}$,
$\boldsymbol{P}'_{i,j,e+1}$ vector from the origin to $P'_{i,j,e+1}$, where $\boldsymbol{P}'_{i,j,e+1} = \boldsymbol{P}''_{i,j,e} + \boldsymbol{T}'_{i,j,e+1}$,
$\boldsymbol{P}''_{i,j,e+1}$ vector from the origin to $P''_{i,j,e+1}$, where $\boldsymbol{P}''_{i,j,e+1} = \boldsymbol{P}''_{i,j,e} + \boldsymbol{T}''_{i,j,e+1}$ and $\boldsymbol{P}''_{i,j,0} = \boldsymbol{P}_{i,j}$,
D length parameter of $\boldsymbol{T}'_{i,j,e}$,
n number of dimensions of the search domain, and
$U''_{i,e}$ objective value at $P''_{i,j}$.

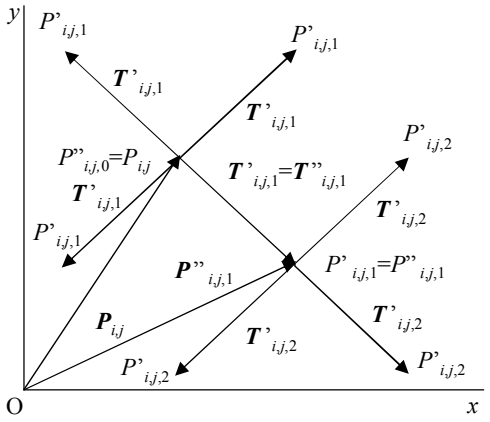

Figure 2. Search points in the neighborhood search routine on a two dimensional X–Y plane

In the PSO search, when $V_{i,j}$ is better than U_i, the algorithm switches to the neighborhood search routine to search better solutions in the neighborhood. In the first step, let $P''_{i,j,0}$ and U_i be $P_{i,j}$ and $V_{i,j}$, respectively. The neighborhood search points are $\boldsymbol{P}'_{i,j,e+1} = \boldsymbol{P}''_{i,j,e} + \boldsymbol{T}'_{i,j,e+1}$ as shown in Figure 2. If r_k = 0 or 1, then $(-1)^{rk}d = d$ or $-d$. Therefore, the number of combinations is 2^n. If e = 0, then the number of search points is 2^n, however, if otherwise, it is 2^n–1. This is because one of the search points, $P'_{i,j,e+1}$, is the same as $P''_{i,j,e}$. Nevertheless, the number of search points is extremely large when n is large. Therefore, we recommend that if the number of search points is greater than a certain number, a limitted number of search points are randomly selected, and solutions are found. The number of the search points and d should be arbitrarily determined according to the type and size of the problem.

In the neighborhood search routine, if $U''_{i,e}$ is better than U_i, then $U_i \leftarrow U''_{i,e}$ and $P_{i,j} \leftarrow P''_{i,j,e}$, and the routine is repeated. Otherwise, the algorithm returns to PSO and continues its search algorithm.

5. NUMERICAL EXPERIMENTS

In order to evaluate the methods described in the section 4, we implemented this method in the standard PSO (this is called the hybrid PSO in this paper) and solved six famous test functions, i.e., Sphere, Rosenbrock, Griewank, Shekel's foxholes, Six hump camel back function and Step function. Those are typical benchmark functions, which are considered to be the minimum standard for performance comparisons of evolutionary algorithms. In this paper, we apply the hybrid PSO to these six test functions and compare its effectiveness with the standard PSO. These functions are used for obtaining the minimum values. The functions used in this paper are as follows;

(a) Sphere

$$f(x) = \sum_{i=1}^{30} x_i^2 \qquad \text{where } -100 \le x_i \le 100 \tag{3}$$

The optimal solution is 0.

(b) Rosenbrock

$$f(x) = \sum_{i=1}^{30} \left[100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2 \right] \tag{4}$$

where $-2.048 \le x_i \le 2.048$

The optimal solution is 0.

(c) Griewank

$$f(x) = 1 + \frac{1}{4000} \sum_{i=1}^{30} x^2 - \prod_{i=1}^{30} \cos\left(\frac{x_i}{\sqrt{i}}\right) \tag{5}$$

where $-600 \le x_i \le 600$

The optimal solution is 0.

(d) Shekel's foxholes

$$f(x) = \left\{ \frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6} \right\}^{-1} \tag{6}$$

where $-65.356 \le x_i \le 65.356$

The optimal solution is 1.

(e) Six hump camel back function

$$f(x) = (4 - 2.1x_1^2 + \frac{x_1^4}{3})x_1^2 + x_1 x_2 + (-4 + 4x_2^2)x_2^2 \tag{7}$$

where $-3 \le x_1 \le 3$, $-2 \le x_2 \le 2$

The Optimal solution is about -1.0316

(f) Step function

$$f(x) = \sum_{i=1}^{5} \lfloor x_i \rfloor \tag{8}$$

where $-5.12 \le x_i \le 5.12$

Optimal solution is -25.

The graphs of these functions are shown in Figure 3

We set a value to w in equation (2) in three different ways.

(i) $w = w_{max} - \dfrac{(w_{max} - w_{min})}{j_{max}} j$

where $w_{max} = 0.9$, $w_{min} = 0.4$, $j_{max} = 500$, $j = 1, 2, \ldots, j_{max}$

We call this w as w_1.

(ii) w: a random number distributed uniformly between 0.4 and 0.9. We call this w as w_2.

(iii) w: a random number distributed uniformly between 0.9 and 1. We call this w as w_3.

We also assign 1 to c_1 and c_2.

Here, we arbitrarily define the number of neighborhood search points as min (2^n-1 (2^n for the first step), n+16). We also set d at 0.05 for these test functions.

100 different initial solutions are generated by random numbers and we solve the problems 100 times using each initial solution. We evaluate the hybrid PSO whether or not it is more effective than the standard PSO.

Table 1 shows the average of 100 solutions up to the 500th iteration by the hybrid PSO and the standard PSO applied to Sphere, Rosenbrock, Griewank, Shekel's foxholes, Six hump camel back function and Step function, when a member size is 50.

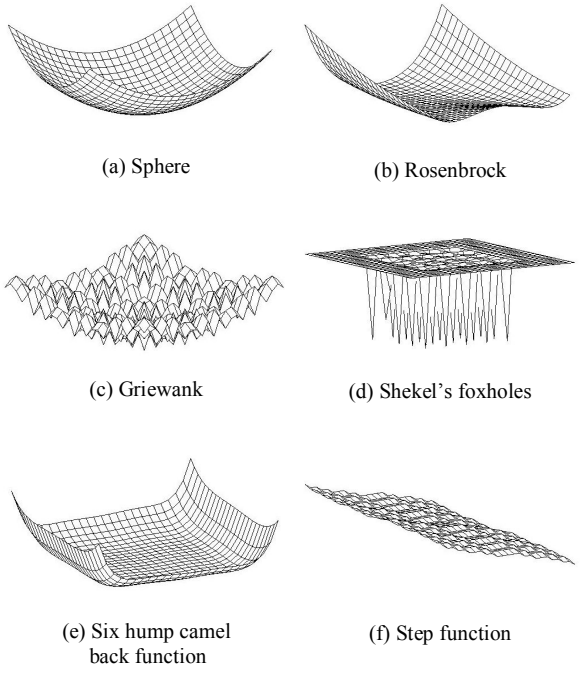

Figure 3. The graphs of the six functions in a three dimensional space

Figures 4(a) through (f) show the results of the hybrid PSO and the standard PSO with the member size of 50 applied to the six test functions under w_2. The hybrid PSO is much better in each w_1, w_2 and w_3 than the standard PSO when applied to Sphere, Rosenbrock and Griewank. In the cases of Shekel's foxholes, Six hump camel back function and Step function, there are no obvious differences. There are also no significant differences among the results of w_1, w_2 and w_3 when applied to the six functions. In Sphere, the obtained minimum values by the hybrid PSO up to the 500th iteration are very close to the optimal solution. In Rosenbrock, the optimal solution is 0 but the hybrid PSO and the standard can not reach the optimal solution up to the 500th iteration; need more iterations to obtain the optimal solution. In Griewank, the obtained minimum values by the hybrid PSO up to the 500th iteration are very close to the optimal solution.

Table 2 shows the number of the minimum values that reached the optimal solutions in Shekel's foxholes, Six hump camel back function and Step function by using the hybrid and the standard PSO when the member size is 50. In Six hump camel back function and Step function, the optimal solution can be obtained until the 35th iteration by the hybrid PSO and until the 50th iteration by the standard PSO. In Shekel's foxholes, about 38 solutions reach to the optimal solution up to the 500th iteration in both PSO.

Figures 5(a) through (f) show the average of 100 solutions in w_2 applied to the six test functions by the hybrid PSO when the member sizes are 10, 30 and 50. It is observed that the member sizes of 30 and 50 derive solutions better than the member size of 10 in Sphere, Rosenbrock and Griewank. However, even if the mem-

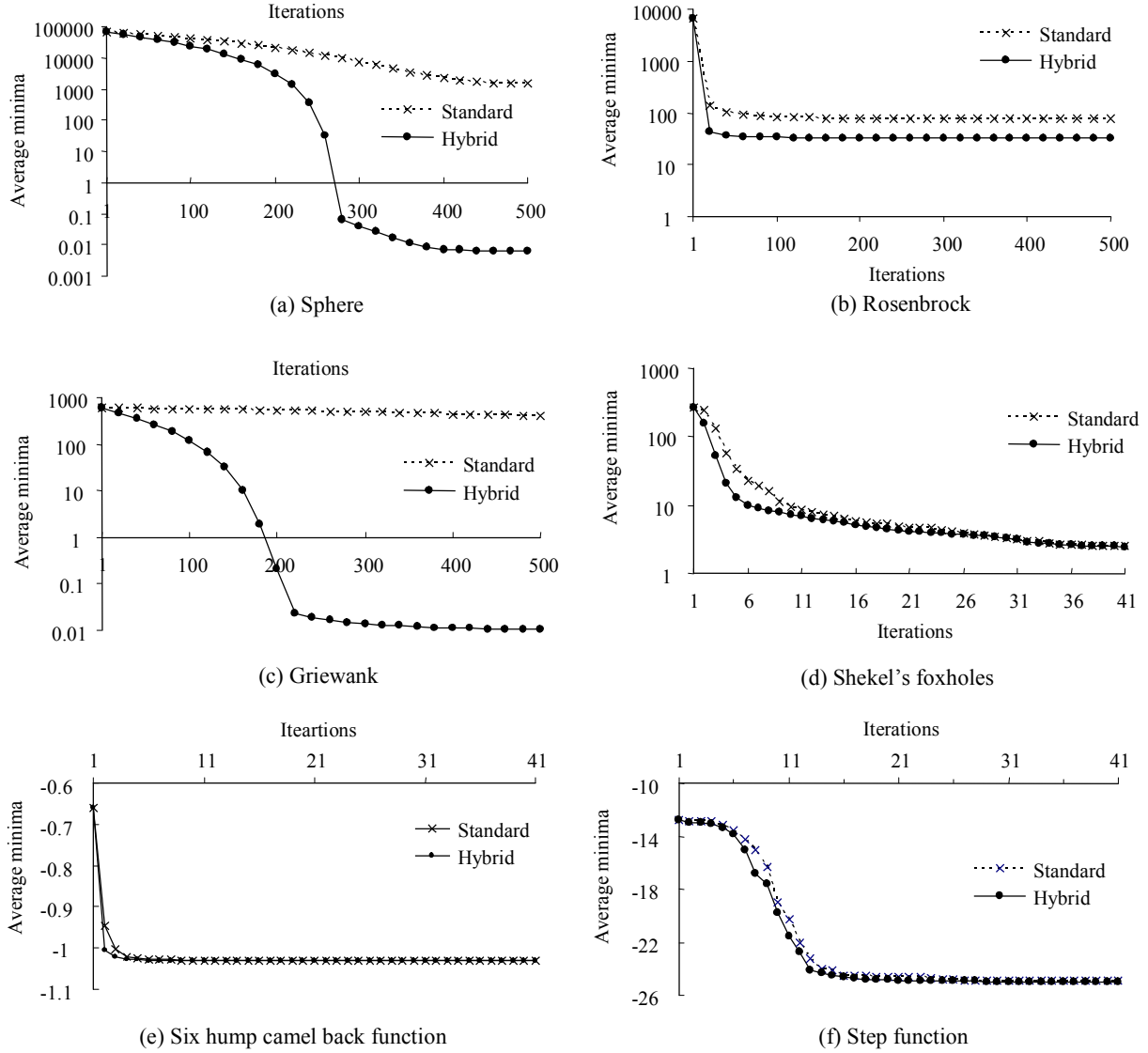

Figure 4. The average of 100 solutions up to 500th iteration by the hybrid PSO and the standard PSO under w_2

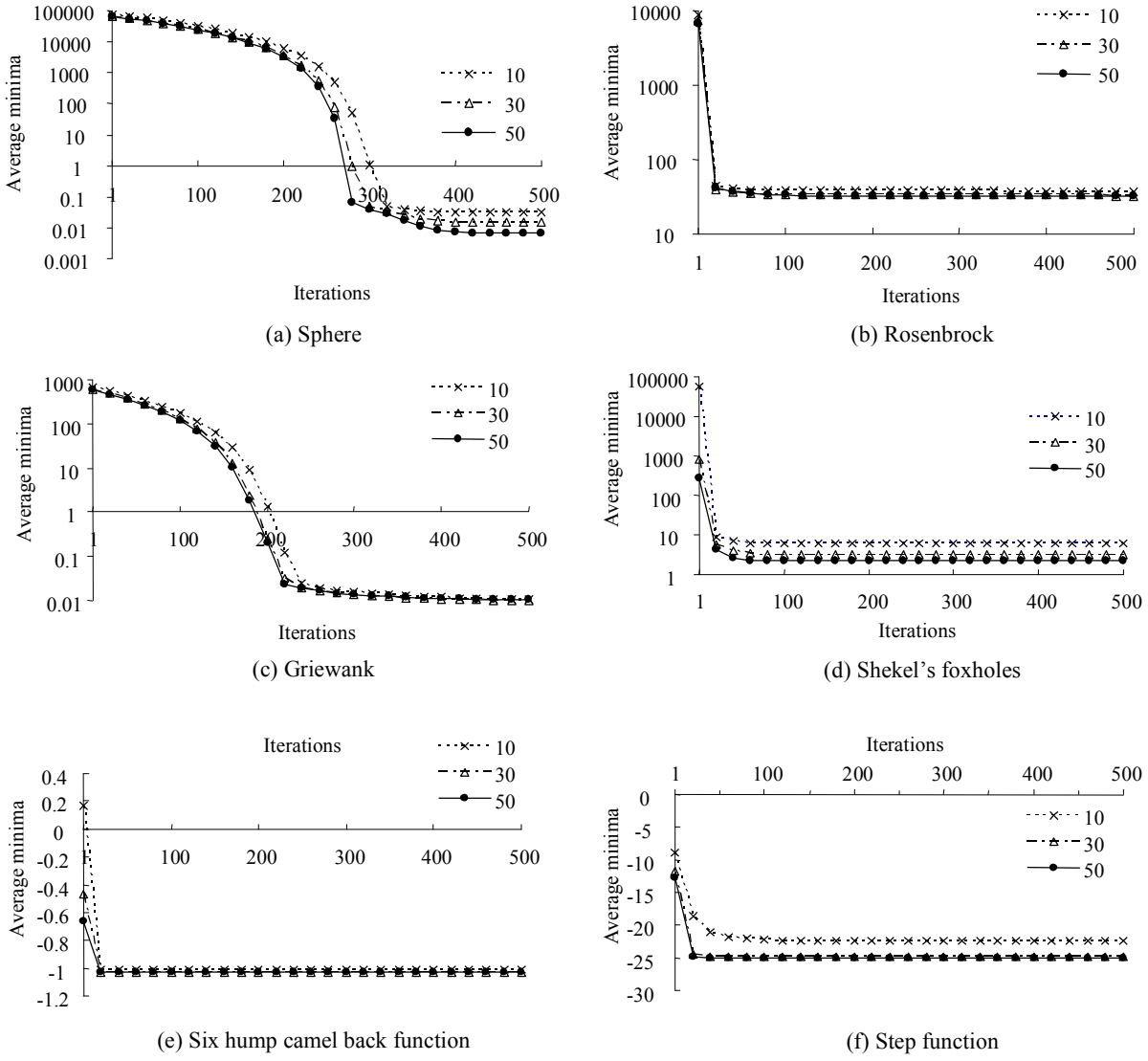

Figure 5. The average of 100 solutions by the hybrid PSO under w_2 when the member sizes are 10, 30 and 50

Table 1. The average of 100 solutions up to 500th iteration

Function	Method	w_1	w_2	w_3
Sphere	Standard PSO	1,876.2	1,529.9	1,979.3
	Hybrid PSO	0.009	0.006	0.008
Rosenbrock	Standard PSO	66.3	77.7	60.9
	Hybrid PSO	32.5	32.6	32.5
Griewank	Standard PSO	409.5	403.3	367.4
	Hybrid PSO	0.011	0.011	0.012
Shekel's foxholes	Standard PSO	2.13	2.20	2.16
	Hybrid PSO	2.22	2.20	2.19
Six hump camel back	Standard PSO	-1.0316	-1.0316	-1.0316
	Hybrid PSO	-1.0316	-1.0316	-1.0316
Step	Standard PSO	-25	-24.9	-25
	Hybrid PSO	-25	-25	-25

ber size is 10, the solutions obtained are good enough. In Shekel's foxholes, Six hump camel back function and Step function, there are significant differences in the average values among the member sizes.

Table 2. The number of solutions that reaches the optimal when the member size is 50

Function	Method	w_1	w_2	w_3
Shekel's foxholes	Standard PSO	37	38	38
	Hybrid PSO	40	38	39
Six hump camel back	Standard PSO	100	100	100
	Hybrid PSO	100	100	100
Step	Standard PSO	100	99	100
	Hybrid PSO	100	100	100

Table 3. The number of solutions that reaches the optimal up to the 500th iteration by the hybrid PSO under w_2 when the member sizes are 10, 30 and 50

Function	10	30	50
Shekel's foxholes	13	28	38
Six hump camel back	98	100	100
Step	85	99	100

Table 3 shows the number of solutions up to the 500th iteration that reaches the optimal solution. From these results, it is obvious that the member size of 30 is enough to reach satisfactorily approximate solutions. In general, as the dimension and the search domain of the function increase, larger member size may derive the better solution faster. Larger member sizes require more computation time. Therefore, member sizes of 30 and 50 are adequate for these kinds of test functions.

Table 4 shows the comparison of computation time and accuracy between the standard PSO and the hybrid PSO. The both PSO were programmed in Digital Visual Fortran, version 6.0 and ran on the computer, Epson Endeavor NT300, Pentium M processor (1.6MHz), 512MB memory with Microsoft Windows XP Professional SP2.

Table 4. Comparison of computation time and accuracy between the standard PSO and the hybrid PSO

Function	Method	Time	Iteration	Value
Sphere	Standard PSO	12.34	5,341	18.537
	Hybrid PSO		500	0.006
Rosenbrock	Standard PSO	0.447	720	74.71
	Hybrid PSO		500	32.61
Griewank	Standard PSO	1.658	1,417	115.51
	Hybrid PSO		500	0.011
Shekel's foxholes	Standard PSO	0.248	574	2.2
	Hybrid PSO		500	2.2
Six hump camel back	Standard PSO	0.0025	30	-1.0316
	Hybrid PSO		30	-1.0316
Step	Standard PSO	0.0039	50	-24.9
	Hybrid PSO		29	-25

(time:sec)

The member size of 50 applied to the six test functions under w_2 in 100 trials. For Sphere, Rosenbrock, Griewank,and Shekel's foshole, the time is the average time period that the iterations of the hybrid PSO repeat 500 times. For the other two test functions, the time is the average time period that the solutions of the hybrid PSO converge to the optimal solution. The values are the average solutions obtained when the procedures of the standard PSO and the hybrid PSO repeat for the time period.

In Sphere, Rosenbrock and Griewank, the solutions by the hybrid PSO are much better than the solutions by the standard PSO for the same computation time. In other three test functions, almost same results can be obtained by both PSO in same computation time. It shows that when the standard PSO cannot easily find a good solution, the neighborhood search routine works very well.

It can be stated that the hybrid PSO is effective for obtaining the better solutions through our numerical experiments. If the number of variables in the problem increases, the computation time will be increase in the hybrid PSO. The hybrid PSO can make the local search with the global search simultaneously. We are sure the hybrid PSO is more effective than the standard PSO.

6. CONCLUSIONS

In this paper, we introduce a neighborhood search algorithm to the standatd PSO. We apply the hybrid PSO to determine the minimum values of the six test functions, i.e., Sphere, Rosenbrok, Griewank, Shekel's foxholes, Six hump camel back function, and Step function. Then, we discuss the performance and effectiveness of the hybrid PSO.

In the hybrid PSO, global search by the standard PSO and local search by the neighborhood search are performed simultaneously. When it is difficult that the standard PSO find a good solution, the neighborhood search routine cooperates with the standard PSO and search in a local area. Finally, the solutions by the hybrid PSO can obtain a better solution than the standard PSO.

Consequently, we show that the hybrid PSO is considerably stable and effective. It can obtain better solutions in less iterations. The hybrid PSO requires more computation time than the standatd PSO, however, both computation times are small enough in practice. Therefore, the hybrid PSO has a possibility to be applied to several types of real-world problems, for example, multi-dimensional knapsack problems, power system control problems, etc.

ACKNOWLEDGMENT

This work was partially supported by JSPS Grants-in-Aid for Scientific Research.

REFERENCES

Clerc, M. (1999), The swarm and the queen: towards a deterministic and adaptive particle swarm optimization, *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1999)*, Washington,

D. C., 1951-1957.

Heppner, F. and Grenander, U. (1990), A stochastic nonlinear model for coordinated bird flocks. In Krasner, S. Ed., The ubiquity of chaos, *AAAS Publications*, Washington, DC.

Kennedy, J. and Eberhart, R. C. (1995), Particle swarm optimization, *Proceedings of the IEEE International Conference on Neural Networks*, *Piscataway*, *NJ*., 1942-1948.

Kennedy, J., Eberhart, R. C., and Shi, Y. (2001), Swarm intelligence, *Morgan Kaufmann Publishers,* San Mateo, CA.

Li, X. (2004), Better Spread and Convergence: Particle Swarm Multiobjective Optimization Using the Maximin Fitness Function, *Proceedings of the Genetic and Evolutionary Computation Conference 2004*, Seattle, Washington, 117-128.

Li, X. (2004), Adaptively Choosing Neighbourhood Bests Using Species in a Particle Swarm Optimizer for Multimodal Function Optimization, *Proceedings of the Genetic and Evolutionary Computation Conference 2004*, Seattle, Washington, 105-116.

Noel, M. M. and Jannett, T. C. (2004), Simulation of a new hybrid particle swarm optimization algorithm, *Proceedings of the Thirty-Sixth Southeastern Symposium on System Theory*, Atlanta, Georgia, 150-153.

Parsopoulos, K. E. and Vrahatis, M. N. (2001), Modification of the Particle Swarm Optimizer for locating All the Global Minima, *Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms*, Prague, Czech Republic, 324-327.

Parsopoulos, K. E. and Vrahatis, M. N. (2002), Initializing the Particle Swarm Optimizer Using the Nonlinear Simplex Method, *Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*, WSEAS Press, Clifton, New Jersey, 216-221.

Reynolds, C. W. (1987), Flocks, herbs, and schools: A distributed behavioral model, *Computer Graphics*, **21**(4), 25-34.

Richards, M. and Ventura, D. (2003), Dynamic Society in Particle Swarm Optimization, *Proceedings of the International Conference on Computational Intelligence and Natural Computing*, Cary, North Carolina, 1557-1560.

Shi, Y. and Eberhart, R. C. (1998), A modified particle swarm optimizer, *Proceedings of the IEEE International Conference on Evolutionary Computation*, Anchorage, Alaska, 69-73.

Toyoda, Y., Yano, F., and Shohdohji, T. (2004), Dynamic Assignment of Parameter Values in Particle Swarm Optimization, *Proceedings of the fifth Asia-Pacific Industrial Engineering and Management Systems Conference*, Gold Coast, Australia, 32.9.1-32.9.10 (in CD-ROM).

Veeramachaneni, K., Peram, T., Mohan, C. K., and Osadciw, L. A. (2003), Optimization Using Particle Swarms with Near Neighbor Interactions, *Proceedings of Genetic and Evolutionary Computation Conference*, Chicago, Illinois, 110-121.

Wilson, E. O. (1975), Sociobiology: The new synthesis, *Belknap Press*, Cambridge, Massachusetts.

Yano, F. and Toyoda Y. (2004), Properties of Particle Swarm Optimization applied to Multi-Hump Camel-Back Functions, *Proceedings of the 33rd International Conference on Computers and Industrial Engineering*, Jeju, Korea, (in CD-ROM).