International Journal of Computer Sciences and Engineering Systems
Vol. 12 No.2 (December, 2018)

New Approaches to Enhance the Computation Efficiency on Particle Swarm Optimization Algorithm

Yang YI[1] and Qiang LI[2]

[1]Computer Science Department, Sun Yat-Sen University GuangZhou, GuangDong 510275, P. R. China
E-mail:issyy@mail.sysu.edu.cn
[2]Computer Science Department, Sun Yat-Sen University GuangZhou, GuangDong 510275, P. R. China
E-mail: liqiang0730@qq.com

Manuscript received January 15, 2018, Manuscript revised March 15, 2018

Abstract: *Methodologies and approaches to improve the computation efficiencies on particle swarm optimization (PSO) is address in the paper. Firstly, several heuristic rules are proposed to be used in particle velocity updating in order to promote their global exploration abilities. Secondly, the theories of building high performance PSO algorithm are described and proved, which consist of introducing a new selection mechanism, inducing a dynamic particle neighbor structure, employing the non-linear inertia weight value and interacting within near neighborhood. Thirdly, a new improved PSO algorithm based upon above theoretical achievements are presented. The notable benchmark tests show that the presented algorithm can obtain the global optimal solutions very quickly. The comparing experiments with two other popular improved PSO algorithms demonstrate that our algorithm has much better performance and lower premature convergence rate than the other methods.*

Keywords: Artificial Life Computing, Particle Swarm Optimization, Dynamic Neighborhood, Velocity Updating, Near Neighbor Interaction

1. INTRODUCTION

PSO (Particle Swarm Optimization) [1], which is inspired by the cognitive behavior of bird flocking or fish schooling, has become a real competitor for GA (Genetic Algorithm) and ACO (Ant Colony Optimization) in evolutionary computation. It has the advantages of simple model, less parameters and relative ease of implementation, which attracts more and more attention of scholars. Nowadays, PSO has been successfully applied to many fields [2-6], such as scheduling, communication, high efficiency computation, E-Business and so on.

PSO can provide high speed of convergence in the initial computation periods, yet, it is apt to trap in a near optimal solution in the later iterations due to its weak global exploration capability. In order to alleviate this problem, Shi and Eberhart[7] introduced an inertia weight into its velocity update equation, which makes the former particle's velocity have a bigger impact on the following particles' velocity changing. Then, they developed the inertia weight linearly decreasing PSO algorithm (IWLD_PSO) [8,9]. In IWLD_PSO, the inertia weight linearly decreased with the iteration and the maximum allowable velocity in each dimension was introduced, which can guarantee the particles search within the feasible solution fields.

Clerk and Eberhart[3] introduced a new concept of constraint factor, which affects three parts of the velocity update parameters, and after that, it will have some influence on the particles 'exploration capability. Another improved PSO with constraint factors is developed(CF_PSO). Shi and Eberhart [9,10] compared IWLD_PSO with CF_PSO in detail. Chatterjee and Siarry[11] provided some general suggestions for parameter selection by using nonlinear variation for inertia weight values.

Although all the modifications mentioned above upgrade the PSO's performance to some extent, there still exits a common challenge that the premature convergence rate remains pretty high[14]. Liu and Wang[12] incorporated the Chaotic Local Search (CLS) into the algorithm by optimizing one fifth of the best particles. Though CLS reduces the premature convergence rate significantly, it cost great number of computation, so, it is not suitable for practical applications.

In this paper, some new principles of building high performance PSO algorithm are brought forward and testified. The new approaches give attention to two or more things, such as local exploitation capability, global exploration capability and convergence speed. A new improved PSO algorithm (NNI_PSO), which adopts a novel selection mechanism, near neighbor interaction method and nonlinearly varying inertia weight value, is described. NNI_PSO owns the characteristic of maintaining more

diversities of the population and moving rapidly towards to the global optimum. The methodologies and approaches may provide some valuable information for solving some complex practical application problem. Benchmark experiments demonstrate that NNI_PSO can achieve the global optimal results with high efficiency, furthermore, the comparative experiments among NNI_PSO, CF_PSO and IWLD_PSO illustrate that NNI_PSO has much better performance than the other two improved PSO algorithms.

The rest of the paper is organized as follows. Section 2 gives a brief description about the methods and algorithm of traditional particle swarm optimization with its all necessary variations. Section 3 analyzes some heuristic methods used in particle velocity updating computation. Section 4 puts forward and testifies some new theories for PSO algorithm optimization. In Section 5, an improved PSO approach is proposed in detail. Benchmark tests and data experiments are performed in Section 6. Conclusion and further efforts are given finally in Section 7.

2. TRADITIONAL PSO ALGORITHM

i, particle;
d, solution space dimension;
v_i, velocity of particle i in feasible solution space, $v_i = (v_{i1}, v_{i2}, \ldots, v_{id})$;
v_{max}, maximum allowable velocity;
x_i, particle i in the searching space; $x_i=(x_{i1}, x_{i2}, \ldots, x_{id})$;
p_i, best position it ever visited so far of particle i, $p_i = (p_{i1}, p_{i2}, \ldots, p_{id})$;
g, index of global optimum position;
p_g, global best position, $p_g = (p_{g1}, p_{g2}, \ldots, p_{gd})$;
x_{max}, limitation of searching range;
k, constraint factor;
w, inertia weight;
n, particles population;
iter, current iteration time;
NC, total iteration times.

In PSO algorithm, each particle is a potential solution for the objective function. Particles adjust their positions combined with the computing process in the iterations, so as to move toward to the global optimum. In a d-dimensional space, each particle is referred to as a point $x_i = (x_{i1}, x_{i2}, \ldots, x_{id})$ in the searching space. Let $v_i = (v_{i1}, v_{i2}, \ldots, v_{id})$ be the velocity of the particle i in the feasible solution space.

Particle i remembers the best position $p_i = (p_{i1}, p_{i2}, \ldots, p_{id})$ it ever visited so fa. The best position in the neighborhood field is represented as $p_g = (p_{g1}, p_{g2}, \ldots, p_{gd})$. At each iteration, particle i adjusts its velocity v_i and position x_i by considering the personal best and the global best ones. The update equations are given as follows [1].

$$v_{id} = wv_{id} + c_1 rand()(p_{id} - x_{id}) + c_2 Rand()(p_{gd} - x_{id}) \quad (1)$$

$$x_{id} = x_{id} + v_{id} \quad (2)$$

c_1 is the cognitive factor and c_2 is the social factor. Generally, let $w = 0.729$, $c_1 = c_2 = 1.4495$. *rand*() and *Rand*() are random number functions and their values are uniformly distributed within [0,1]. w is the inertia weight, who reflecting the control and influence degree of current particle to its next generation or neighbors. w will impact both of the velocity and position on the particles.

Pseudo-code of PSO_Algorithm

Step 1: Initialize n, NC, d, k, w, c_1 and c_2; initialize v_i and x_i stochastically;
Step 2: For particle (i = 1 to n), do
{
Step 3: Calculate f_i // fitness of particle i;
Step 4: Find the p_g and its index g;
// global optimal particle
Step 5: Let $p_i = x_i$;
}
Step 6: While (Stop criterion is not satisfied), do
{
Step 7: Update x_i and v_i of each particle i
// by using Eqs. (1) and (2)
$v_{id} = wv_{id} + c_1 rand()(p_{id} - x_{id}) + \{c_2 Rand()(p_{gd} - x_{id})$;
$x_{id} = x_{id} + v_{id}$;
Step 8: Recalculate f_i of particle i;
Step 9: Update p_i for each particle;
Step 10: Update g and p_g ;
}
Step 11: Stop.

3. INTELLIGENCE RULES ON PARTICLE VELOCITY UPDATING

It can be noticed in Eq. (1) that v_{id} is composed of three elements: (1) the influence from previous particles' velocity; (2) the cognitive from other particle; (3) social affection from the particle environment.

Heuristic Rule 1 (HR1)[7]: It will be better for the PSO algorithm to restrict particles velocities in a field scale, which means $v_{id} \in [-v_{max1}, v_{max2}]$ for all the particles, $v_{max1} >= 0$; $v_{max2} >= 0$

Based up our researches on the properties of evolutionary computation and also inspired by other methods, we present several new heuristic rules to control the particle velocity and the movement distance, in order to increase the algorithm's calculation efficiency.

Heuristic Rule 2 (HR2)[8]: there exists an allowable range for particles' movements in the searching space for each iteration, which means the distance of each movement is between $[x_{max1}, x_{max2}]$, $x_{max1} >= 0$, $x_{max2} >= 0$.

Heuristic Rule 3 (HR3): let x_{max2}, the maximal allowable movement range of particles in one step, to be bigger in the earlier period than it is in the later computation period.

Heuristic Rule 4 (HR4): let x_{max1}, the minimal movement range for particles in one step, to be bigger in the

earlier period than it in the later period during the computation.

Heuristic Rule 5 (HR5): let v_{max2}, the maximal particle velocity, to be bigger in the earlier period than it in the later period in the process of iteration computation.

Heuristic Rule 6 (HR6): let v_{max1}, the minimal particle velocity, to be smaller in the earlier period than it in the later period in the process of iteration computation.

The particles move in the fields of vector space, so, there exist negative velocities. For any iteration searching based intelligent algorithm, the early computation focus on the global searching and exploration abilities in the early period, and then, pay special attention on the convergence speed in the final calculation period. With a bigger velocity and moving distance limitation, the particles can have a bigger chance to move quickly in a larger space, in order to find out the global optimal solution. On the other hand, in the final period of computation, a smaller velocity and moving distance limitation will prevent the particle moving out of the feasible space, and then, make the particles concentrate to the global optimal point quickly.

IWLD_PSO Inertia Weight Rule (IWLD_Rule)[8]: the inertia weight descends linearly with the increasing of iteration times.

IWLD_Rule is illustrated by Eq. (3). w_{final} and $w_{initial}$ are the predetermined maximum and minimum inertia weight values, respectively.

$$w = (w_{final} - w_{initial})(NC - iter) / NC + w_{initial} \quad (3)$$

A large inertia weight value facilitates a global search while a small inertia weight facilitates a local search [9]. By linearly decreasing, the inertia weight value changes from a relatively larger value to a smaller one through the process of computing, so that, the algorithm tends to have more global search abilities at the beginning period while more local search abilities in the final period.

So, a proper selection on the inertia weight value w_{final} and $w_{initial}$ can make a balance between the exploitation capability and the exploration capability.

CF_PSO Constriction Factor Rule (CF_Rule)[10]: adding a new factor to restrict the velocity, so as to limit the maximum velocity v_{max} to a dynamic range.

CF_Rule is illustrated by Eq. (4) and Eq. (5) .

$$v_{id} = k[\, v_{id} + c_1 r\, and\, ()(p_{id} - x_{id}) + c_2 Rand()(p_{gd} - x_{id})] \quad (4)$$

$$k = 2 \Big/ \left| 2 - \varphi - \sqrt{\varphi^2 - 4\varphi} \right|, \varphi = c_1 + c_2, \varphi > 4 \quad (5)$$

The influence factor k will have impact on each of the three components in velocity update equation. CF_Rule can be considered as a special case of HR1 to HR6, whose objective 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, such as let $v_{max}=x_{max}$.

Generally, the parameters in above formulas are given by $c_1=c_2=2.05$, and $k=0.729$, equivalent to the case of setting.

4. OPTIMAL METHODOLOGIES TO IMPROVE PSO ALGORITHM

With the help of above intelligent methodologies, the improved PSO algorithm shows a good performance at the beginning of the computation, and also maintains a relatively high speed of convergence speed near the end calculation. However, it has no way to deal some other big challenges, such as adjusting the iteration searching direction dynamically according to the acquired information and escaping premature convergence.

The major cause of premature convergence is due to the decreases on the diversity character of the particles since the information exchanges very rapidly. We develop a new approach, named by *new Near Neighbor Interaction* based PSO (NNI_PSO), which can maintains a high speed exchange of information among particles, and can also let the algorithm maintain a rather high-level diversity of particle swarm.

Remark 1: Preserving some fittest particles during the whole computing process will abate the premature convergence problem for the PSO algorithm.

Proof: Keeping some fittest particles can eliminate the chance of selecting unsuitable particles[13]. Generate new ones stochastically for the population *n*, ust like infusing some "flesh blood" continually flowing into the swarm, the swarm diversity will be kept at a high level during all the process of computation.

High level swarm diversity can effectively alleviate the premature convergence problem of PSO. NNI_PSO owns the ability of alleviating the premature convergence problem. Remark1 is used as a new selection mechanism of PSO algorithm.

Definition 1: near neighbor, each particle and its four following ones consist of a near neighbor.

Remark 2: Inducting current best particle from the near neighbor field into the velocity updating can improve the global exploration ability near the end of the calculation.

Proof. The new velocity updating expression is shown in Eq. (6) and Eq. (7), each particle and its four following particles constitute a neighborhood.

$$v_{id} = w\, v_{id} + c_1 rand()(p_{id} - x_{id}) + c_2 * rand()*(p_{id}* - x_{id}) + c_3 rand()(p_{gd} - x_{id}) \quad (6)$$

$$x_{id} = x_{id} + v_{id} \quad (7)$$

p_{id}* is the best in near neighborhood; c_2* is the near neighbor's social factor; *rand*()* is the random function for the new added element.

It is clear that, by inserting a new element into the updating equation, the current particles will have a greater chance to cluster toward the direction of the neighborhood optimal position to interact with the best particle in near neighborhood.

The convergence rate of NNI_PSO becomes much larger. So, the problem of lacking of global exploration ability near the end of the calculation is corrected.

Remark 3: Introducing a power factor into the inertia weight calculation may have chance to enhance the computation efficiency.

Proof: As show in Eq. (8), the new inertia weight updating equation is modified by introducing a power factor *m*, then, the *w* will change non-linearly along with the computing.

$$w^* = (w_{final} - w_{initial})[(NC - iter) / NC]^m + w_{initial} \quad (8)$$

$\because$ ß0<$(NC - iter) / NC$<=1;

$\because$ the value of $[(NC - iter) / NC]^m$ is a positive decimal fraction which becomes smaller and smaller while the computing processing.

$\because$ In the earlier computing period, $w^* \approx w$, but in the later to the end period, w^* will be more smaller than w.

We change the expression of Eq. (6) to Eq. (9), and Eq. (7) to Eq. (10), under the principle of Remark 3.

$$v_{id}\,*= w^*\, v_{id} + c_1 r\, and()(p_{id} - x_{id}) + c_2 * rand()*(p_{id}\,* - x_{id}) + c_3 rand()(p_{gd} - x_{id}) \quad (9)$$

$$x_{id}\,*= x_{id} + v_{id}\,* \quad (10)$$

$\therefore$ At the beginning computing period, $v_{id}\,* \approx v_{id}$, $x_{id}\,* \approx x_{id}$, when closing to the end, $v_{id}\,* << v_{id}$, $x_{id}\,* << x_{id}$.

The initial value of w^* is relative small. It will be non-linearly reduced to a small value through the course of the PSO running. According to the HR1-HR6, Remark 3 will make the algorithm to have larger global search ability at the beginning computation phase and also help the algorithm to avoid leaving out of the solution fields in the later period.

$\therefore$ $x_{id}\,*$ is better than x_{id}. Introducing the power factor *m* may bring a better chance to enhance the algorithm's efficiency.

5. STEP-BY-STEP PROCESS OF NNI_PSO

In NNI_PSO algorithm, the velocity update equation is composed of four parts, which makes "the process of particles learning" more similar to that of nature. Remark 1, Remark 2 and Remark 3 have some interior relationships. Remark 1 determines which particle is worthy of learning and which should be abandoned; Remark 2 provides new neighbors to the current particle, so as to assure the particles have continuous learning object; and Remark 3 presents a dynamically searching method for particles. The benefits from above remarks result in that the active particles in the population are those who can be optimized continuously, and then the state of swarm environment will stay healthy sustained.

Pseudo-code of NNI_PSO

Step1: Initialize *n*, *NC*, v_{max}, *d*, x_{max}, *k*, *w*, c_1, c_2, c_2*, *m*; initialize v_i and x_i stochastically; *near neighbor* size;

Step 2: For particle (*i*=1 to *n*), do

{

Step 3: Calculate f_i , // fitness value of particle *i*;

Step 4: Find p_g and its index *g*;

// the global optimal particle

Step 5: Let $p_i = x_i$;

}

Step 6: While (Stop criterion is not satisfied), do

{

Step 7: Calculate w^* by using Eq. (8)

$w^* = (w_{final} - w_{initial})[(NC - iter) / NC]^m + w_{initial}$

Step 8: Update x_i and v_i of each particle *i*

// by using Eq. (9) and Eq. (10)

$v_{id}\,*= w^*\, v_{id} + c_1 rand()(p_{id} - x_{id}) + c_2 * rand()*(p_{id}\,* - x_{id}) + c_3 rand()(p_{gd} - x_{id})$

$x_{id}\,*= x_{id} + v_{id}\,*$

Step 9: Recalculate f_i of particle *i*;

Step 10: Update p_i for each particle;

Step 11: Update the neighbor optimum p_{id}* all particles;

Step 12: Replace one-fifth worst performance particles in the swarm population with new generated ones;

Step 13:Update *g* and p_g ;

}

Step 14: Stop.

6. EXPERIMENTS AND ANALYSIS

6.1 Benchmark Functions

NNI_PSO algorithm is applied to five notable benchmark functions to evaluate its computing efficiency and its results quality. The five benchmark functions f_0 to f_4 are given in Eq. (11) to Eq. (15)

$f_0(x) = \min$ £$_{i=1}{}^n x_i^2$,

where $x = [x_1, x_2, x_n]$, is the real number vector (11)

$f_1(x) = \min \Sigma_{i=1}{}^n (100(x_{i+1} - x_i)^2 + (x_i - 1)^2)$ (12)

$f_2(x) = \min \Sigma_{i=1}{}^n (x_i^2 - 10\cos(2nx_i) + \{10)$ (13)

$$f_3(x) = \min(20 + e - 20e^{-0.2\sqrt{\frac{\sum_{i=1}^n x_i^2}{n}}} - e^{\frac{\sum_{i=1}^n \cos 2\pi x_i}{n}}) \quad (14)$$

$f_4(x) = \min (1/n)\Sigma_{i=1}{}^n (x_i^4 - 16x_i^2 + \{5x_i)$ (15)

The global optimal values of the functions from f_0 to f_3 are all 0, and that of f_4 is -78.3323. Function f_4 is considered to be very complex and difficult to be optimized, since its local optimal solutions increase exponentially with the increase of dimension.

6.2 Experimental Results and Analysis

(a) Setting the Parameters' Values

In the calculation of NNI_PSO, the size of neighborhood is set to be 5; the neighborhood includes the particle itself and its four offspring. In order to test the effectiveness of the presented algorithm, data tests are processed.

The space dimension of the benchmark is [10, 20, 30, 40, 60], respectively. The value of v_{max} and x_{max} are set beforehand randomly and provided in Table 1.

Table 1
v_{max} and x_{max} of each Funtion

function	v_{max}	x_{max}
f_0	5.12	5.12
f_1	2.048	2.048
f_2	10	10
f_3	32	32
f_4	100	100

The parameters used in those algorithms are established according to the experiences from lots of experiments and the instances form other research literatures. The comparative experiments are performed among NNI_PSO, CF_PSO and IWLD_PSO, presented in Table 2.

Table 2
Parameters Value in Different Algorithms

	c_1	c_2	c_2*	k	m	w_{final}	$w_{initial}$	n
CF_PSO	2.05	2.05	—	0.729	—	—	—	
IWLD_PSO	2	2	—	—	—	0.9	0.4	20/40
NNI_PSO	1	1	2	—	1.2	0.9	0.2	

(b) Comparative Experiments on Benchmark Functions

Detail results are shown in Table 3 and Table 4. In Table 3, the final output is the average optimal solution after 50 iterations, and the maximum iteration is 100 times of the problem dimension.

For function f_0, NNI_PSO performs better than CF_PSO and IWLD_PSO except that when the dimension is 30.

For f_1 and f_2, the results of NNI_PSO is much better than that of CF_PSO and IWLD_PSO, the optimal solution of NNI_PSO is almost the real global optimal while neither CF_PSO nor IWLD_PSO could achieve a satisfied solution.

For function f_3, when the scale of the problem becomes larger than 10, NNI_PSO will obtain much better than the other two algorithms.

For the complex multi-modal function f_4, NNI_PSO can obtain the global optimum solution while CF_PSO and IWLD_PSO can never.

(c) Larger Scale Problems Experiments

In order to testify that the performance of NNI_PSO is much better than both of IWLD_PSO and CF_PSO when there are much more particles and the dimension is very big, another experiment is addressed shown in Table 4. It can be noticed that NNI_PSO performs better than CF_PSO and IWLD_PSO in most of the cases.

Table 3
Comparative Experiments of NNI_PS, CP_PSO and IWLD_PSO (population = 20)

			Fitness value			
Funct.	*dimension*	*iteration*	*IWLD_PSO*	*CF_PSO*	*NNI_PSO (n=1.2)*	*NNI_PSO (n=1)*
	10	1000	6.025 E-15	3.788 E-40	3.555 E-54	8.589 E-48
f_0	20	2000	5.530 E-4	5.374 E-11	1.070 E-27	1.104 E-30
	30	3000	5.975 E-4	4.497 E-8	6.019 E-15	5.253 E-9
	10	1000	10.789	2.576	0.004	0.011
f_1	20	2000	13.357	18.515	0.011	0.074
	30	3000	42.592	41.168	0.007	0.018
	10	1000	1.989	1.989	8.587 E-7	1.801 E-4
f_2	20	2000	10.952	11.939	9.257 E-4	0.002
	30	3000	22.884	42.783	7.172 E-5	3.253 E-4
	10	1000	0.002	0.003	0.006	0.002
f_3	20	2000	7.771 E-4	5.839 E-4	9.177 E-4	0.006
	30	3000	0.001	0.001	5.172 E-4	2.001 E-4
	10	1000	-71.735	-72.678	-72.678	-78.3323
f_4	20	2000	-72.678	-75.504	-78.332	-78.3323
	30	3000	-73.62	-74.562	-75.504	-78.3323

Table 4
Comparative Experiments on Large Scale Problems (population = 40)

function	*dimension*	*iteration*	*IWLD_PSO*	*CF_PSO*	*NNI_PSO*
f_0	20	2000	1.122 E-23	8.543 E-26	4.963 E-63
	40	4000	9.609 E-24	8.508 E-25	1.873 E-34
	60	6000	6.524 E-20	1.784 E-23	3.522 E-25
f_1	20	2000	3.815 E-25	5.406 E-26	1.609 E-24
	40	4000	4.940 E-15	1.577 E-19	2.649 E-22
	60	6000	3.461 E-15	7.618 E-22	1.100 E-22
f_2	20	2000	1.433 E-23	8.985 E-26	1.085 E-25
	40	4000	7.467 E-23	8.957 E-22	3.326 E-26
	60	6000	9.935 E-27	1.010 E-24	4.397 E-25
f_3	20	2000	1.322 E-23	1.083 E-21	2.489 E-16
	40	4000	1.561 E-23	1.461 E-19	4.241 E-26
	60	6000	8.307 E-23	2.208 E-21	1.011 E-22
f_4	20	2000	-69.85	-74.091	-73.774
	40	4000	-73.62	-72.678	-78.332
	60	6000	-71.264	-72.206	-75.735

(d) Comparative Experiments with Different Factor Values

Some other comparison experiments with different factor values are made as proposed in Fig.1 and Fig.2. Set the power factor of NNI_PSO *m* by 1, make *w* non-linearly decrease. Differ from IWLD_PSO, NNI_PSO keeps the near neighbor interaction. The experiment results are presented in the last column of Table 4. NNI_PSO's results are better than IWLD_PSO and CF_PSO.

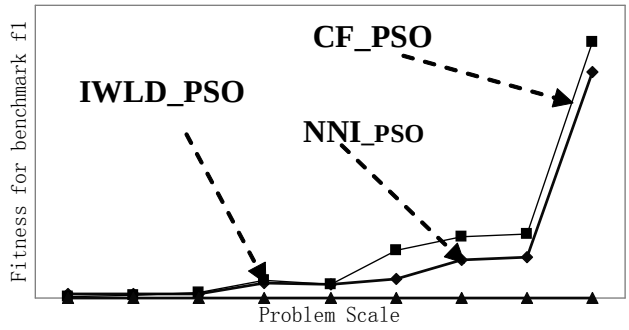

Figure 1: Results of f_1 of the Algorithms

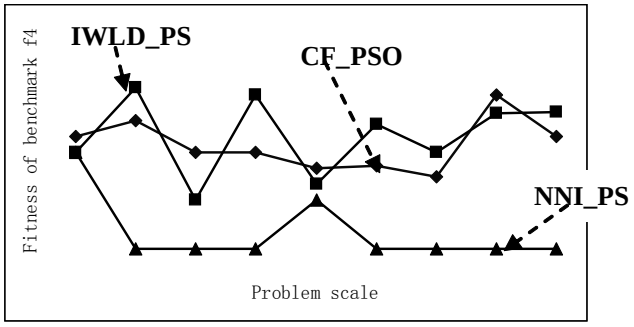

Figure 2: Results of f_4 of the Algorithms

7. CONCLUSION AND FUTURE RESEARCHES

The results in Table 3 and Table 4 indicate that NNI_PSO has a bigger chance to achieve the real global optimal results than the other improved PSO algorithms. Fig 1 and Fig 2 demonstrate that NNI_PSO can convergence with higher accuracy. The practical data tests suggest that the mechanism of near neighbor interaction may help easing the premature convergence effectively while maintaining rapid convergence rate. So, the developed method in the NNI_PSO can be considered as a successful and effective improvement for PSO based algorithm.

The contributions of the paper can be summarized as follows.

(a) Several new heuristic rules to limit the maximal and minimal values of particle' velocity and its one-step movement range are established. Then, both of the global search abilities and the computing effectiveness have been enhanced.

(b) A new selection mechanism (Remark 1) is presented and proved, by which the premature convergence during the computation is decreased.

(c) The principles of using near neighbor optimal particle (Remark 2), and combining with the nonlinearly varying inertia weight (Remark 3), are addressed and proved. The presented algorithm adjusts the neighborhood structure of particles dynamically using neighbor interaction, and those methods are of great help to global searching.

(d) The step-by-step process about the new improved PSO algorithm is described in detail. Benchmark function experiments, together with the comparative experiments with two other popular PSO based

algorithm, demonstrate that the presented algorithm (NNI_PSO) possesses the advantages of rapid convergence speed and high-level diversity of the swarm. A high level diversity helps easing the premature convergence problem, and near neighbor interaction guides the particles moving toward global optimum effectively.

However, there is still some chance for the improvement in NNI_PSO. Our researches in the future will focus on applying NNI_PSO in more practical decision problem, so as to fully verify its efficiency and accuracy.

ACKNOWLEDGMENTS

This paper is supported by the National Natural Science Foundation of China (60573159).

REFERENCES

[1] J. Kennedy, and R. C. Eberhart, "Particle Swarm Optimization", in Proceedings of IEEE International Conference on Neural Networks, Perth, Australia, IEEE, 1995, **4**, 1942-1948.

[2] F. V. D. Bergh, and A. Engelbrecht, "Particle Swarm Weight Initialization in Multi-layer Perception Artificial Neural Networks", in Development and Practice of Artificial Intelligence Techniques, Durban, South Africa, 1999, 41-45.

[3] M. Clerc, and J. Kennedy, "The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Complex Space", in *IEEE Transactions on Evolutionary Computation*, **6**, 2002, 58-73.

[4] J. L. Ching, T. T. Chao, and L. Pin, "A Discrete Version of Particle Swarm Optimization for Flowshop Scheduling Problems", in Computers & Operations Research, Elsevier, **34**, 2005, 3099-3111.

[5] F. Elizabeth, G. Gouvea, and C. G. Marco, "Particle Swarm for the Traveling Salesman Problem", in Evolutionary Computation in Combinatorial Optimization, Springer Berlin / Heidelberg, **3906**, 2006, 99-110.

[6] Z. Hong, L. Heng, and C. M. Tam, "Particle Swarm Optimization-based Schemes for Resource-Constrained Project Scheduling", in Automation in Construction, **14**, 2005, 393-404.

[7] Y. Shi, and R. C. Eberhart, "A Modified Particle Swarm Optimizer", in IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, May 4-9, 1998, 69-73.

[8] Y. Shi, and R. C. Eberhart, "Parameter Selection in Particle Swarm Optimization", in Evolutionary Programming VII, Lecture Notes in Computer Science, Springer Berlin / Heidelberg, **1447**, 1998, 591-600.

[9] Y. Shi, and R. C. Eberhart, "Empirical Study of Particle Swarm Optimization", in Congress on Evolutionary Computation, Washington DC, USA, July 6-9, 1999, **3**, 1945-1950.

[10] Y. Shi, and R. C. Eberhart, "Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization", in Proceedings of 2000 Congress Evolutionary Computation, La Jolla, CA, USA, 2000, **1**, 84-88.

[11] A. Chatterjee, and P. Siarry, "Nonlinear Inertia Weight Variation for Dynamic Adaptation in Particle Swarm Optimization", in Computers & Operations Research. Elsevier, **33**, 2006, 859-871.

[12] B. Liu, L. Wang, Y. H. Jin, F. Tang, and D. X. Huang, "Improved Particle Swarm Optimization Combined with Chaos", in Chaos Solitons & Fractals, Elsevier, **25**, 2005, 1261-271.

[13] R. Jacques, and S. V. Jakob, "A Diversity-guided Particle Swarm Optimizer –the ADPSO" available: *http://citeseer.nj.nec.com/riget02diversityguided.html*, April 11, 2007.

[14] S. M. Arvind, M. Rui, W. Christopher, and P. Christian, "Neighborhood Re-structuring in Particle Swarm Optimization", in AI 2005: Advances in Artificial Intelligence, Springer Berlin / Heidelberg, **3809**, 2005, 776-785.