Adaptive Control of Acceleration Coefficients for Particle Swarm Optimization Based on Clustering Analysis

Zhi-hui ZHAN, Jing Xiao, Jun ZHANG, *Member, IEEE* and Wei-neng Chen

***Abstract*—Research into setting the values of the acceleration coefficients c_1 and c_2 in Particle Swarm Optimization (PSO) is one of the most significant and promising areas in evolutionary computation. Parameters c_1 and c_2 in PSO indicate the "self-cognitive" and "social-influence" components which are important for the ability to explore and converge respectively. Instead of using fixed value of c_1 and c_2 with 2.0, this paper presents the use of clustering analysis to adaptively adjust the value of these two parameters in PSO. By applying the *K*-means algorithm, distribution of the population in the search space is clustered in each generation. An adaptive system which is based on considering the relative size of the cluster containing the best particle and the one containing the worst particle is used to adjust the values of c_1 and c_2. The proposed method has been applied to optimize multidimensional mathematical functions, and the simulation results demonstrate that the proposed method performs with a faster convergence rate and better solutions when compared with the methods with fixed values of c_1 and c_2.**

I. Introduction

OPTIMIZATION problems are widely encountered in various fields of science and technology. So far as long, a number of optimization algorithms have been proposed [1] , such as Genetic Algorithm [2](GA), Simulated Annealing[3] (SA), Evolutionary Programming [4](EP), Artificial Neural Network[5] (ANN), Ant Colony Optimization[6] (ACO), etc. Traditional optimization algorithms as above are proved to be efficient in handling many classes of optimization problems.

As another evolutional optimization method, Particle Swarm Optimization (PSO) was first introduced by Kennedy and Eberhart in 1995[7][8]. PSO is inspired by the behavior of bird flocking and fish schooling, and has developed fast in recent years, major due to its simple concept and easiness for implementation. PSO is a population-based, generation iterative algorithms like genetic algorithm, however, it is different that PSO uses only two equations (see equations (1) and (2)) to update each particle's velocity and position generation by generation instead of the selection, crossover and mutation operations which are essential in GAs[9].

In PSO, a swarm of particles are introduced to represent the potential solutions, and each particle is associated with two vectors, the velocity $V_i = \left[v_i^1, v_i^2, \cdots, v_i^D \right]$ vector and the position $X_i = \left[x_i^1, x_i^2, \cdots, x_i^D \right]$ vector, where D means that the solution is in D-dimension space. In the initialization, the velocity and position of each particle are set randomly within the velocity limitation and search space respectively. During the iterations, the present position of the particle will be calculated by a fitness function as the merit (fitness value), if the fitness is better than the fitness of $pBest_i$, which stores the best solution that the i^{th} particle has explored so far, then the $pBest_i$ will be replaced by the current solution (include the position and fitness). At the same time of selecting $pBest_i$, the algorithm selects the best $pBest_i$ of the swarm as the global best, which is regarded as *gBest*. Then, the velocity and position of each particle will be updated using the following two equations:

$$v_i^d = \omega * v_i^d + c_1 * r_1^d * (pBest_i^d - x_i^d) + c_2 * r_2^d * (gBest^d - x_i^d) \quad (1)$$

$$x_i^d = x_i^d + v_i^d \quad (2)$$

Where ω is considered as the inertia weight[10]; c_1, c_2 are known as acceleration coefficients that are traditionally set as the fixed values 2.0; r_1^d and r_2^d are two separately generated uniformly distributed random numbers in the range [0, 1] for the d^{th} dimension[7]. After updating the velocity and position, iteration goes on until the stop criterion is met.

Notice that there was no inertia weight ω in the equation (1) when PSO was first proposed in 1995[8], but the positive variant $V_{\max}^d$ was used to clamp the maximum velocity of each particle on the d^{th} dimension, so if the update velocity $\left| v_i^d \right|$ exceeds the value $V_{\max}^d$ which was specified by the users, v_i^d will be assigned as $sign(v_i^d) V_{\max}^d$. By using the inertia weight ω, the $V_{\max}^d$ can be eliminated [11]. Although the maximum velocity can be clamped by $V_{\max}^d$ or ω, however, the update position yielded by equation (2) should be carefully considered during the running period, each time we update the position of each particle, if the new position is out of the potential range of the problem, resetting has to be done. Some researchers reset the exceeded x_i^d to $X_{\min}^d$ if it negatively exceeded, or $X_{\max}^d$ if it positively exceeded, and some researchers reset x_i^d to a random value between $X_{\min}^d$

This work was supported in part by NSF of China Project No.60573066 and NSF of Guangdong Project No. 5003346

Zhi-hui ZHAN, Jing Xiao, Jun zhang and Wei-neng Chen are with the Department of Computer Science, SUN Yat-sen University, Guangzhou, China.

Jun ZHANG is corresponding author, e-mail: junzhang@ieee.org .

and $X_{\max}^{d}$, however, in this paper, we mix using these two reset modes, by fixing to the bound or to a valid random value stochastically.

The general description of this algorithm is given as follows:

Step 1: Initialization*:* A population of particles (popular size is between 20 and 40), are initiated with random positions and velocities. The position and velocity vectors of each particle are with the same dimension as the problem dimension.

Step 2: For each particle *i*, measure the fitness with current position, and replace the $pBest_i$ if the current fitness is better than $pBest_i$. What is more, the global best *gBest* will be also replaced by $pBest_i$ if $pBest_i$ is better than *gBest*.

Step 3: For each particle in the swarm, update its position and velocity by the two equations showed as (1) and (2). And the restriction of search range should be checked.

Step 4: Repeat step 2)-3) until termination criterion is satisfied. For example the maximal iteration comes or the error threshold is met.

For the simple concept of PSO, it has been introduced into many applied fields for optimization [11]and many researchers have done lots of work to improve the algorithm.

The linear decreased inertia weight ω was first proposed by Y. Shi and Eberhart[10], (show as equation (3)). This method can adjust the global search ability and local search ability.

$$\omega = \omega_{\max} - (\omega_{\max} - \omega_{\min}) * \frac{gen}{GENERATION} \quad (3)$$

What is more, M.Clerc [12] introduced a constriction factor to PSO. The constriction factor modified the former equation (1) to equation (4) and (5).

$$v_i^d = K*[v_i^d + c_1 * r_1^d * (pBest_i^d - x_i^d) + c_2 * r_2^d * (gBest^d - x_i^d)] \quad (4)$$

$$K = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}, where \quad \varphi = c_1 + c_2, \quad \varphi > 4 \quad (5)$$

The value of *K* was set to 0.729, and c_1, c_2 were both 2.05.

Another active research trend is to combine PSO with other evolutionary computation techniques, and this made up variants of hybrid PSOs. Angeline [13]used the selection operation like GA in PSO, and the hybridization of GA and PSO was also been used in [14]for recurrent network design.

Topology structure of PSO is also studied by researchers and different topologies have been proposed to improve the traditional PSO. The study by Kennedy [15] shows that small neighborhoods might work better on complex problems, while larger neighborhood is better for simple problems. In order to avoid the drawback of fixed neighborhood, the dynamically changing neighborhood structure was proposed by Hu and Eberhart[16].

Although PSO algorithm has been used into many fields and variants of improved approaches have been proposed, the easiness of trapping into the local optima is still the major deficiency of this technology. By using clustering analysis, this paper aims to adaptively control the acceleration coefficients in PSO, which is expected to keep the diversity of the swarm to escape the local optima as well as to obtain faster convergence rate.

The rest of the paper will be organized as follows. In Section II, a novel PSO with adaptive control of acceleration coefficients will be proposed. Then in Section III, numerical experiments results with fixed/adaptive-control acceleration coefficients are compared. At last, in section IV, conclusion is summarized.

II. Adaptive Control of C_1 and C_2

A. Behavior of Parameters c_1 and c_2

PSO is simple because each particle uses only two equations to update the velocity and the position. So, acceleration coefficients c_1 and c_2 are considered as important factors for the success of PSO. Parameter c_1 pulls the particle toward its own best location, this behavior can enhance the groping ability, keep the diversity of the swarm. The role of parameter c_2, however, performs as the convergent factor that draws all the particles toward to the global best. It can be imagined that the values of c_1 and c_2 should be adjusted in different evolutional states, for example, larger c_1 and smaller c_2 should be set in the initialize state to enhance the explored ability and keep the diversity, whilst c_1 should be reduced and c_2 should be increased for converging to the best solution if the swarm is in the matured state. All these adjustment rules are discussed in Section II-D.

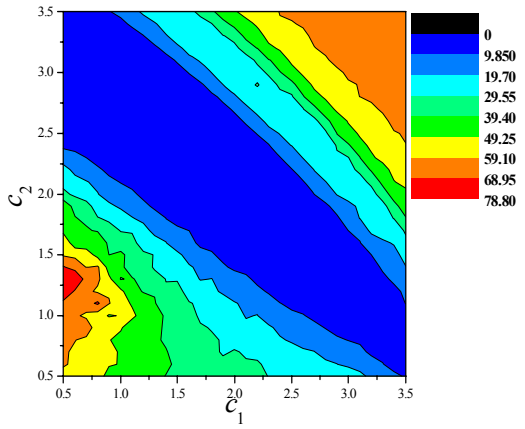
This paper presents the use of clustering analysis to adaptively tune c_1 and c_2. By applying the *K*-means algorithm [17], the distribution of the population in the search space is clustered in each generation. The values of c_1 and c_2 are adjusted adaptively by considering the relative size of the cluster containing the best particle and the one containing the worst particle. The method suggests the use of the relative population distribution to define the training states and has been used in GA for adaptive control crossover and mutation probability and resulted in faster converge rate and high solution accuracy [2], as another evolutional algorithm like GA, PSO also encounters different evolutional states and different values for acceleration coefficients can be used in different phases for better performance.

B. Investigation of Parameters c_1 and c_2

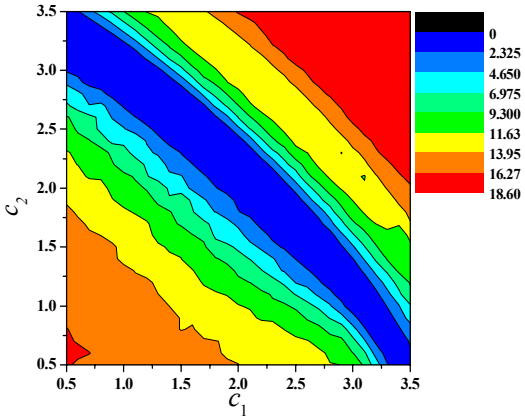
In order to get an insight of how c_1 and c_2 influence the performance of PSO and gain the value ranges of them for getting considerable solution, this paper first gives the investigated conclusion about parameters c_1 and c_2. We let c_1 and c_2 span from 0.5 to 3.5 separately and observe the results of each couple of c_1 and c_2. 100 trials are carried out for each test function with each couple of combination (all the test function are presented in Table II and described in Section III-A), and the mean values make up the contour plot figures. These figures can clearly indicate the performance of each couple of c_1 and c_2, and make it clear that what values should be set and what bounds should be limited of these two parameters for better performance. As the contour plot figures are almost the same, we just show 2 figures that contain the

unimodal function as f_1 and multimodal functions as f_3. The figures are showed as Fig. 1.

The figures indicate that the sum of c_1 and c_2 should be



(a) Schwefel's function f_1



(b) Ackley's function f_3

Fig. 1. The contour plot of functions for different couples of c_1 and c_2 spanning from 0.5 to 3.5 separately, the higher the color in the color indicating bar, the better (smaller) value it represents.

clamped between 3.0 and 4.0 as well as that each of them should be set between 0 and 4 itself, if we want to get better results. This conclusion also points out that commonly using of fixed values 2.0 is one of the good choices. This investigation conclusion will be used in the adaptive control parameters adjusting system proposed in this paper to limit the values of c_1 and c_2.

C. Clustering of the Population

The *K*-means algorithm is sufficient for the particular application to depict the particle distribution although it can only partition sub-optimal clusters [17]. Assume that the population is partitioned into *K* clusters. The clustering process described as follows.

Step 1: Create *K* initial cluster centers CP^1, CP^2, …, CP^K randomly for the *K* clusters C_1, C_2, …, C_K.

Step 2: Assign P_n, 1<=*n*<=*POPSIZE*, which is the particle in the swarm to cluster C_j, where $j \in \{1, 2, \ldots, K\}$ if and only if

$$\left\| P_n - CP^j \right\| < \left\| P_n - CP^p \right\|, \quad 1<=p<=K, \text{ and } j \neq p \quad (6)$$

where $\left\| P_n - CP^j \right\|$ is the distance between P_n and CP^j.

Step 3: Compute new cluster centers $CP^{1,*}$, $CP^{2,*}$, …, $CP^{K,*}$ as follows:

$$CP^{j,*} = \frac{1}{M_j} \sum_{P_n \in C_j} P_n, \quad 1<=j<=K \quad (7)$$

Where M_j is the number of elements belonging to cluster C_j.

Step 4: If $CP^{j,*} = CP^j$, 1<=*j*<=*K*, the process will be terminated, and CP^1, …, CP^K are chosen as the cluster centers. Otherwise, assign $CP^j = CP^{j,*}$, 1<=*j*<=*K*, and *step 2* will be started again.

This paper divides the population into three clusters, and after the clustering process described as above, the size of the cluster that contains the global best particle is set as G_B, and the size of the cluster containing the global worst particle is considered as G_W. At the same time, the largest cluster G_{max} and the smallest cluster G_{min} can also be determined.

D. Tuning Rules for c_1 and c_2

Tuning of c_1 and c_2 in the proposed method is based on considering the relative cluster sizes of G_B, G_W, G_{max} and G_{min} (i.e. $G_B = G_{max}$). Four rules for tuning the values of c_1 and c_2 are defined; and details are described as follows and are tabulated in Table I.

TABLE I
RULES DEFINED FOR TUNING THE VALUES OF C_1 AND C_2

Rules	Conditions	States	c_1	c_2
Rule1	$G_B=G_{min}$, $G_W= G_{max}$	Initial	Increase	Reduce
Rule2	$G_B= G_W=G_{min}$	Submature	Reduce	Reduce
Rule3	$G_B= G_W= G_{max}$	Maturing	Increase	Increase
Rule4	$G_B=G_{max}$, $G_W=G_{min}$	Matured	Reduce	Increase

Rule 1 - The best particle is in the smallest cluster and the worst particle is in the largest cluster.

The training process is considered to be in the initial state. The diversity of the swarm should be kept to explore optima as many as possibly. A relative larger c_1 together with a relative smaller c_2 can make the individual move toward its own best position rather than the global best for diversity. This metaphor persuades us to increase the value of c_1 and reduce the value of c_2.

Rule 2 - G_B equals G_W. Both of them equal the smallest cluster G_{min}.

The training process is considered to be in the sub-mature state. The values of c_1 and c_2 are reduced. Consider that both the global best particle and the global worst particle are in the

smallest cluster, we have less information about what status the swarm is going to form, it is wised to let the particles move freely rather than force them toward the *pBest* or *gBest*.

Rule 3 - G_B equals G_W. Both of them equal the largest cluster G_{max}.

The training process is considered to be in the maturing state. The effect of *gBest* should be enhanced to guide the swarm converge; however, the probability of premature is also necessary to be avoided, so the value of c_1 should be increased to keep the diversity. Hence, both values of c_1 and c_2 are increased in this training state.

Rule 4 - The global best particle is in the largest cluster whilst the global worst particle is in the smallest cluster.

The training process is considered to be in the matured state. The value of c_1 is reduced whilst the value of c_2 is increased in this training phase. In such a training state, most of the particles with similar component vectors as *gBest* have swarmed together, and the *gBest* is most possibly the solution of the optimization problem. The relative larger c_2 together with the relative smaller c_1 is propitious to local search surrounded the *gBest*, and better solution is more likely gained by this learning mechanism.

It is crucial to note that the any decision to the above consideration should not lead to brute-stop or brute-force c_1 or c_2, otherwise, the philosophy of evolutionary computation will be lost. Hence, some limitations should be used while changing the values of c_1 and c_2. Section II-E discusses the details of adaptive tuning mechanism for c_1 and c_2.

E. Adaptive Tuning Mechanism for c_1 and c_2

Tuning of c_1 and c_2 are based on an adaptive tuning system. The process is to convert the inferred evolutional state to a crisp value. The input of the system is the cluster size gained by clustering analysis, G_B, G_W, G_{max} and G_{min}. The inter process is based on the judgment rules within the input variants. The output of the system, however, is the change of c_1 and c_2. Thus, the actual change of c_1 and c_2 is determined by adding $c_1(gen-1)$ and $c_2(gen-1)$ to the output change $K_i\delta(i=1,2)$. They are showed as (8) and (9):

$$c_1(gen) = c_1(gen-1) + K_1\delta \quad (8)$$

$$c_2(gen) = c_2(gen-1) + K_2\delta \quad (9)$$

Where δ is used to keep the change of c_1 and c_2 within a tolerance percentage of the level in each generation, the early experiments indicated that a random value in the interval [0.01, 0.1] performed better on most of the test functions; K_1 and K_2 are chosen to decide the addition or subtraction of the value, which are compatible with Table I. For example, if the training state is the initial state, K_1 is positive and K_2 is negative obeying the presentation in Table I.

However, it should be noted that c_1 and c_2 have limitations. For example, as the investigation conclusion presented in Section II-B, this paper uses the limitation of that $0 \le c_1, c_2 \le 4.0, 3.0 \le c_1 + c_2 \le 4.0$, if the sum is smaller than 3.0, both of c_1 and c_2 are magnified, and if the sum is larger than 4.0, both of c_1 and c_2 are shrinked.

III. Experiments and Comparisons

Experiments based on mathematical benchmarks functions are simulated in this section. In order to demonstrate the advantages of the proposed approach, variants PSOs are used for comparison, experiments results and discussions are presented.

A. Mathematical Functions and Variants PSOs

This paper uses 4 typic mathematical functions as listed in Table II for experiment which all have 0 as their minimal value. These test functions are widely used to test the evolutionary algorithms [18]. f_1 is Schwefel's function, it is a unimodal function and f_2, f_3, f_4 are known as Rastrigrin, Ackley and Generalized Penalized functions respectively, they are all multimodal functions. All these four functions are difficult for optimization, especially the multimodal functions which are all complex problems with many local optima. The local optima increase exponentially as the dimension increases, in order to show the efficient of the proposed method, a high dimension as 30 is used.

TABLE II
THE TEST FUNCTIONS FOR COMPARISON.
DETAILS OF F_4 IS GIVEN IN APPENDIX

Test functions	*SD*	*Error*
$f_1(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	$[-10,10]^{30}$	0.01
$f_2(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	$[-5.12,5.12]^{30}$	125
$f_3(x) = -20\exp(-0.2\sqrt{1/n\sum_{i=1}^{n} x_i^2}) - \exp(1/n\sum_{i=1}^{n}\cos 2\pi x_i) + 20 + e$	$[-32,32]^{30}$	0.1
$f_4^* = \frac{1}{10}\{\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i-1)^2[1+\sin^2(3\pi x_{i+1})] + (x_n-1)^2[1+\sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i,5,100,4)$	$[-50,50]^{30}$	0.1

Variants of PSOs are used here for comparison. The first type PSO is the one proposed in [10], referred as IWPSO (Inertia Weight PSO) here, the second one is referred as CFPSO (Constriction Factor PSO) introduced in [12], and the third one is the adaptive acceleration coefficients method proposed in this paper, referred as APSO (Adaptive PSO). As the proposed parameters settings and commonly used settings, IWPSO starts with a $\omega_{\max} = 0.9$ and ends with a $\omega_{\min} = 0.4$, and the inertia weight ω obeys the equation (3) during the running period. In the CFPSO, the constriction factor K in equation (4) is set as 0.729 and the acceleration coefficients c_1 and c_2 are both 2.05. The parameters of APSO proposed in this paper, however, the inertia weight ω is set to a fixed value 0.5, which can get a better trade-off between the global search and local search. We don't adopt the time decreasing ω as IWPSO because the adaptive tuning of c_1

and c_2 can balance the explode ability and exploited ability well. c_1 and c_2 in APSO are set as 2.0 in the beginning and adaptively adjust during the running.

Another notice is the computation efforts of these three PSOs. Although APSO brings in a clustering operation before the update operation, nevertheless, the computation effort of the clustering analysis is much lighter when is compared with the fitness evaluation of the being solved problem. In fact, due to the small population size of PSO, for example, 20 particles in the swarm, the clustering analysis can terminate in no more than 5 loops in each generation.

For the fair test of all the three PSOs, they share the same population size of 20 and 2000 training generations for each test function. On the purpose of avoiding stochastic error, each function is simulated for 100 trials and the mean values are calculated and recorded.

B. Results Comparison and Discussion

Table III lists the solutions gained by each PSOs for different test functions. The boldface is used in the table to indicate the best results among the three methods. In order to illustrate the performance of these algorithms, Fig. 2 presents the comparison results by showing the evolutional curves on different problems with the three PSOs methods respectively in the same figure.

These comparison results show that among the IWPSO, CFPSO and APSO, APSO can obtain a much better solution in all the four test function.

The results of f_1 indicate that APSO can outperform other methods in solving unimodal function. However, the most advantaged feature of APSO is that it can jump out of the local optima and get much better solutions while optimizing the multimodal functions. The mean solutions and variation of the 100 trials for different test functions optimizing by the three PSOs presented in Table III indicate that the proposed APSO outperforms the traditional methods with not only higher quality solutions but also steadier solutions. What is more, the figures about the Rastrigin's function, Ackley's function and Generalized Penalized's functions (f_2, f_3, f_4) in Fig. 2 illustrate that APSO can refine the solution better and better as the evolutional generations increasing whilst other algorithms are stagnated in the local minima.

TABLE III
RESULTS COMPARISON FOR VARIANTS PSOS

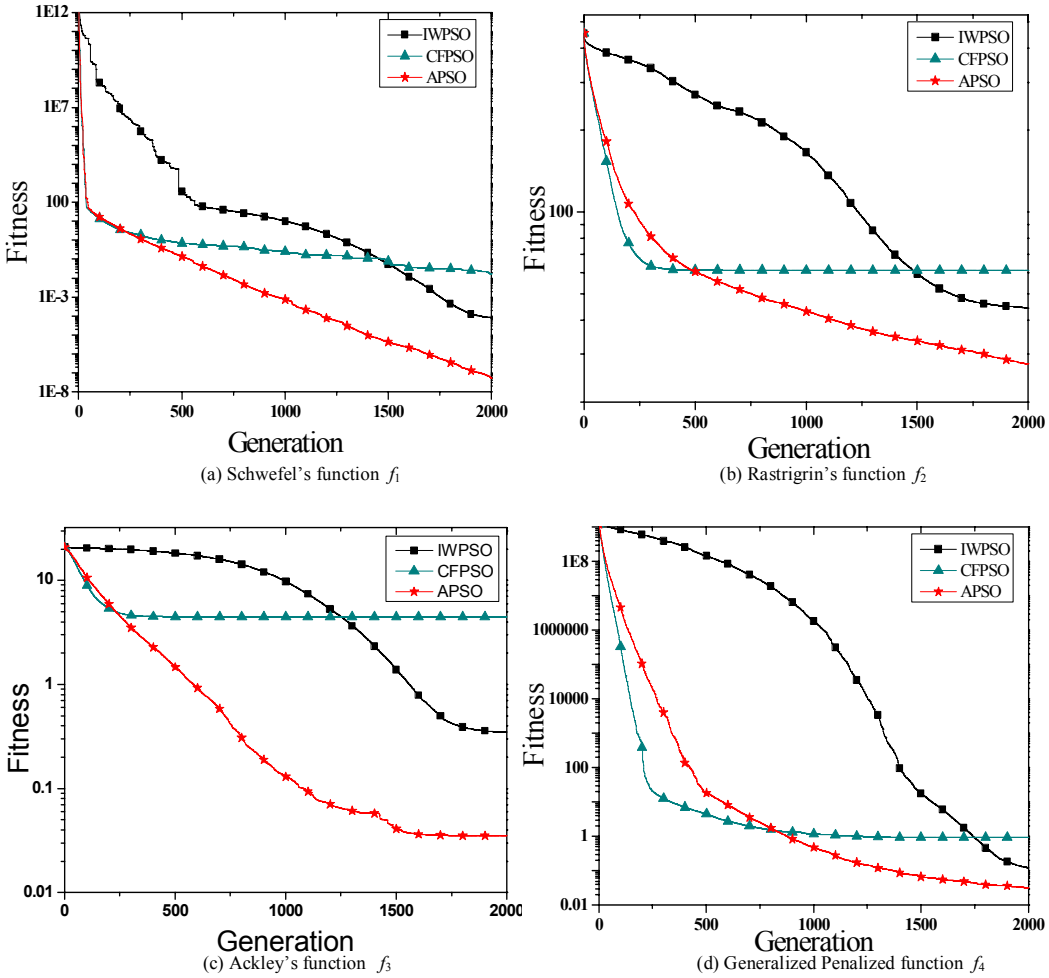
functions		IWPSO	CFPSO	APSO
f_1	Mean	8.104E-05	1.643E-02	**5.398E-08**
	Var	2.295E-07	4.270E-03	**5.867E-14**
f_2	Mean	4.433E+01	6.097E+01	**2.760E+01**
	Var	1.241E+02	2.492E+02	**1.033E+02**
f_3	Mean	3.471E-01	4.451E+00	**3.519E-02**
	Var	3.867E-01	4.098E+00	**6.247E-02**
f_4	Mean	1.188E-01	9.519E-01	**3.212E-02**
	Var	1.948E-01	2.565E+00	**3.290E-03**

Boldface is used to indicate the best results.

TABLE IV
THE NUMBER OF TRIALS THAT OBTAIN THE *ERROR* (THE ACCEPTED ACCURACY OF EACH FUNCTION) WITH SPECIALLY GENERATIONS (1000, 1500 AND 2000 RESPECTIVELY)

Generation \ Functions		f_1	f_2	f_3	f_4
1000	IWPSO	0	8	0	0
	CFPSO	51	**100**	0	**40**
	APSO	**99**	**100**	**88**	33
1500	IWPSO	0	**100**	1	0
	CFPSO	77	**100**	0	48
	APSO	**100**	**100**	**97**	**83**
2000	IWPSO	**100**	**100**	75	79
	CFPSO	87	**100**	0	48
	APSO	**100**	**100**	**98**	**92**

Boldface is used to indicate the best results.



(a) Schwefel's function f_1 (b) Rastrigrin's function f_2

(c) Ackley's function f_3 (d) Generalized Penalized function f_4

Fig. 2. The mean fitness in every generation for each test function

Not only the proposed APSO can avoid premature and get better solution accuracy while optimizing multimodal functions, but also it has a faster convergence rate while compared with the other algorithms in most of the test functions. In order to give a comparison of the convergence rate among the PSOs, Fig. 3 presents number of trials which obtains the solution in every generation while solving f_1. Three normal curves are generated for IWPSO, CFPSO and APSO respectively. The curve is generated using the mean and standard deviation of the counts number, and is normalized to the largest bin value of the histogram. It is obviously that if the curve is toward to the left side, it means that most of the trials can determine the solution in the early generation, hence Fig. 3 indicates that APSO can determine the solution fastest among the three test PSOs. What is more, the cumulative frequency of solutions obtained in each generation in solving f_1 is illustrated in Fig. 4. Additionally, Table IV benchmarks the searching speed of optimizing all the test functions listed in Table II. This table can give a comparison of the number of trials that can determine the solution after 1000, 1500, and 2000 generations respectively. For instance, while optimizing f_1, IWPSO can only get the solutions after 1500 generation. However, 51 trials, 77 trials and 87 trials out of the 100 trials have found the solution after 1000 generations, 1500 generations and 2000 generations respectively in CFPSO. On the other hand, in the proposed APSO in this paper, 99 out of the 100 simulations can determine the solution only after 1000 generations, all of the simulations can find the solution if the generation is larger than 1500, or 2000.

By comparing the results of IWPSO, CFPSO, and APSO, we can see that the proposed adaptive control of acceleration coefficients is efficient for PSO. APSO in this paper outperforms the other algorithms in most of the test functions with the faster convergence rate and escape the local optima to obtain better optimization results.

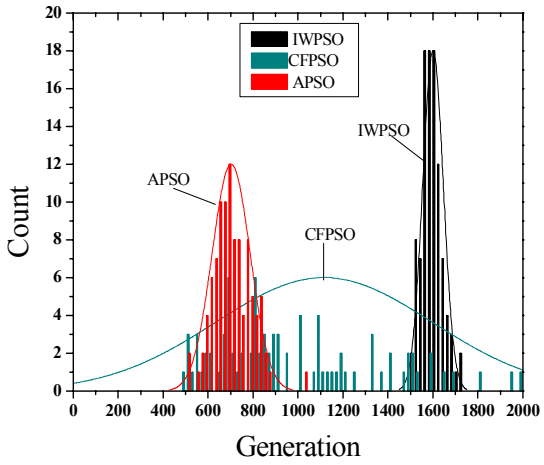


Fig. 3. Count of solution (*Error*) obtained in each generation in different PSOs for f_1.

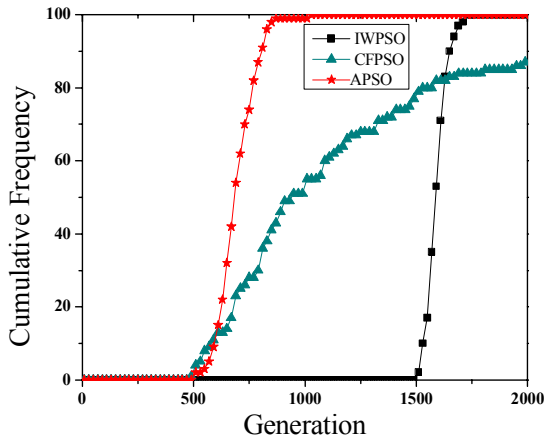


Fig. 4. Cumulative frequency of the solutions (*Error*) obtained in each generation in different PSOs for f_1.

IV. Conclusion

A clustered-based control parameters c_1 and c_2 in PSO has been proposed. It is in the manner that the values of c_1 and c_2 are adapted to the population distribution of the solutions which indicates the evolutional states. Simulations based on mathematics benchmark functions have been carried out with the previous proposed PSOs with the fixed c_1 and c_2 and the one proposed in this paper respectively, the results show that the proposed method outperforms the previous methods with not only faster convergence rate, but also higher quality of solutions. In the future work, further investigation of evolutional states with special parameters tuning rules will be carried out and more test functions will be used for the comparison.

Appendix

* Details of f_4:

$$u(x_i,a,k,m)=\begin{cases} k(x_i-a)^m, x_i > a \\ 0, -a \le x_i \le a \\ k(-x_i-a)^m, x_i < a \end{cases}$$

References

[1] X. Yao, Ed., "Evolutionary Computation: Theory and Applications," *Singapore: World Scientific*, 1999.

[2] J. Zhang, H. S.-H. Chung and W.-L, Lo "Clustering-Based Adaptive Crossover and Mutation Probabilities for Genetic Algorithms," *IEEE Transactions on Evolutionary Computation* : Accepted for future publication. Volume PP, Issue 99, 2006 Page(s):1 -1.

[3] S.Kirkpatrick, "Optimization by Simulated Annealing," *Science*, vol.220, no.4598, pp.671-680, May 1983.

[4] L. J. Fogel, "Evolutionary programming in perspective: The top-down view," in *Computational Intelligence: Imitating Life*, J. M. Zurada, R. J.Marks II, and C. Goldberg, Eds. Piscataway, NJ: IEEE Press, 1994.

[5] Hopfield J .J. "Neural networks and physical system with collective computational abilities," *Proc Natural Acadimic Science* USA, 1982, 79(3):2554-2558.

[6] Macro Dorigo, Vittorio Maniezzo and Alberto Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Trans. on systems man*, and cybernetics - *part B: cybernetics*, vol. 26, 1996, pp. 29-41.

[7] J. Kennedy and R.C. Eberhart, "Particle Swarm Optimization," in *Proc. IEEE Int. Conf. Neural Networks*. 1995, vol. 4, pp. 1942–1948.

[8] R.C. Eberhart and J. Kennedy, "A New Optimizer Using Particle Swarm Theory," in *Proc. 6th Int. Symp. Micro Machine and Human Science*, 1995, pp 39-43.

[9] Y. Shi and R. C. Eberhart, "Comparison between genetic algorithms and particle swarm optimization," in *Lecture Notes in Computer Science—Evolutionary Programming* VII, vol. 1447, Proc. 7th Int. Conf. Evolutionary Programming, Mar. 1998, pp. 611–616.

[10] Y. Shi and R. Eberhart, "A Modified Particle Swarm Optimizer," in *Proc.IEEE World Congr. Comput. Intell,* 1998, pp. 69–73.

[11] R.C. Eberhart. and Shi, Y. "Particle Swarm Optimization: Developments, Applications and Resources," *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2001)*, Seoul, Korea. 2001.pp.81-86.

[12] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability and convergence in a multidimensional complex space," *IEEE Trans. Evol. Comput*., vol. 6, no. 2, pp. 58-73, Feb. 2002.

[13] P. Angeline, "Using selection to improve particle swarm optimization," in *Proc. IJCNN'99*, Washington, DC, July 1999, pp. 84–89.

[14] Chia-Feng Juang, "A Hybrid of Genetic Algorithm and Particle Swarm Optimization for Recurrent Network Design," *IEEE Trans. System, man, and Cybernetics*, vol. 34, no. 2, April 2004, pp. 997-1006.

[15] J. Kennedy and R. Mendes, "Neighborhood topologies in fully informed and best-of-neighborhood particle swarms," *IEEE Trans. Sys. Man, and Cybernetics Part C.*, vol.36, no.4, pp. 515-519. July. 2006.

[16] X. Hu and R. C. Eberhart, "Multiobjective optimization using dynamic neighborhood particle swarm optimization," in *Proc. Congr. Evol. Comput*., Honolulu, HI, 2002, pp. 1677–1681.

[17] J. T. Tou, R. C. Gonzalez, *Pattern Recognition Principles*, Reading MA: Addison-Wesley, 1974.

[18] X. Yao, Y. Liu, and G. M. Lin, "Evolutionary programming made faster," *IEEE Trans. Evol. Comput*., vol. 3, no. 2, pp. 82–102, Jul. 1999.