Adaptive Constriction Factor for Location-related Particle Swarm

XIANG-HAN CHEN, WEI-PING LEE, CHEN-YI LIAO, JANG-TING DAI
Institute of Management Information System
Chung Yuan Christian University
200, Chung-Pei Rd., Chung-Li, Tao-Yuan 32023, Taiwan
R.O.C.
http://chenxianghan.googlepages.com

Abstract: -Particle Swarm Optimization (PSO) has received increased attention in the evolutionary computation fields recently. In the paper, we proposed Adaptive constriction factor for Location-related Particle Swarm (ALPS) that is shown to be superior when compared with the existing PSO algorithm. We adapt a technique of overcoming complex problems with PSO. This is accomplished by using the ratio of the relative location of better particles to determine the direction in which each constriction factor of the particle needs to be varied. Finally, we are presented experiment results on benchmark functions testify ALPS's efficiency.

Key-Words: - Particle swarm optimization, optimization, evolutionary computation, constriction factor, adaptive method.

1 Introduction

The Particle Swarm Optimization (PSO) is a population based search optimization technique. Compared with evolutionary computation techniques, such as simulated annealing (SA), evolutionary algorithms, genetic algorithms (GA), and ant colony optimization (ACO), on the one hand they are motivated by the evolution [12] with the search for an optimum; and are iterate-based process that is based on random decisions in the search space [8], and on the other PSO was based on the simulation of simplified animal social behaviors such as fish schooling, bird flocking, etc. In PSO, a member in the swarm, called a particle, represents a feasible solution which is a point in the search space. At each generation, this alogrithm to find the global best solution by simply varying the trajectory of each particle toward its own experience best position and toward the best particle of the entire swarm [9]. This technique is becoming very popular due to simplicity of implemention and ability to quickly converge to a reasonably excellent solution [11]. However, the higher dimension problems usually have more complicated search spaces, where particles may become trapped more easily in local minima.

The PSO imitates the swarm behavior and the individuals represent points in the *N*-dimension search space, where *N* is number of the parameters to be optimized. Each particle represents a feasible solution. In the PSO algorithm, the trajectory of each individual in the search space is adjusted by dynamically varying the velocity of each particle, according to its own flying experience and flying experience of the other members in the search space [9]. A swarm comprises the horde of particles that fly through the potential solution space to explore and exploit optimal solutions. Each particle updates its position based on its own best exploration, best swarm overall experience, and its previous velocity vector according to following model:

$$v_{id} = w \times v_{id} + c_1 \times \varphi_1 \times \left(p_{id} - x_{id} \right) + c_2 \times \varphi_2 \times \left(p_{gd} - x_{id} \right) \quad (1)$$

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

In PSO, the coordinates of each particle represent a conceivable solution joint with two vectors, the position (x_i) and velocity vector (v_i). In N-dimension search space, $x_i = \left[x_{i1}, x_{i2}, x_{i3}, ..., x_{iN} \right]$ is the position of the *i*th particle and $v_i = \left[v_{i1}, v_{i2}, v_{i3}, ..., x_{iN} \right]$ is a function the velocity vectors associated with each particle *i*. $p_i = \left[p_{i1}, p_{i2}, p_{i3}, ..., p_{iN} \right]$ represents the best previous position (the position giving the best fitness value) of the *i*th particle and $p_g = \left[p_{g1}, p_{g2}, p_{g3}, ..., p_{gN} \right]$ represents the experience of the most successful particle. Each particle updates its position based on equation (1), the first, second, third terms, represent its previous velocity vector v_{id} , its best previous position vector $\left(p_{id} - x_{id} \right)$, and best position discovered by the whole population vector $\left(p_{gd} - x_{id} \right)$; its new position according to the following updating equation (2). Since the PSO, several improvements have been suggested. It is a common modification of the basic PSO algorithm to linearly decrease the value of parameter *w* over time. The function of inertia weight is to balance global exploration p_{gd} and local exploitation p_{id} . The addition of the linearly decreasing inertia weight results in faster convergence [4]. This is done to adjust the swarm's behavior form exploration of the whole search space to exploitation of probable regions. Through empirical studies, Eberhart and Shi have found the optimal solution can be improved by shifting the value of *w* from 0.9 to 0.4 at the search for most problems. The mathematical repesentation of this concept is given by (3).

$$w = \left(w_{\max} - w_{\min} \right) \times \frac{\left(MAXITER - iter \right)}{MAXITER} + w_{\min} \quad (3)$$

where w_{max} and w_{min} are the initial and final values of the inertia weight. *MAXITER* is the maximum allocated number of iterations and *iter* is the current iteration number.

The constriction is implemented as in the other version; this improvement as introduced by Clerc. They show that the constriction PSO can converge without using Vmax. By the constriction coefficient, the amplitude of the particle's oscillation decreases, resulting in its convergence over time [2]. The PSO with equation (4) is called Particle Swarm Optimization with Constriction Factor (PSO-cf).

$$v_{id} = \kappa\left(w \times v_{id} + c_1 \times \varphi_1 \times \left(p_{id} - x_{id}\right) + c_2 \times \varphi_2 \times \left(p_{gd} - x_{id}\right)\right) \quad (4)$$

$$\kappa = \frac{2}{\left|2 - \phi - \sqrt{\phi^2 - 4\phi}\right|} \quad (5)$$

and $\phi = c_1 + c_2$, $\phi > 4$.

Considering these concerns, Eberhart and Shi have Comparing Inertia Weight and Constriction Factors. They discovered that Constriction Factors *k* was better Convergence than inertia weight [5]. Further, Clerc and Kennedy showed an excellent reference that analyzed and studied the PSO promising convergence characteristics. They had established some mathematical foundation to explain the behavior of a simplified PSO model in its search for an optimal solution [3].

This paper combines adaptive alogrithm to the PSO, resulting in novel collaborative PSO model, namely Adaptive constriction factor for Location-related Particle Swarm (ALPS). The ALPS model is between elitist particles' position and oneself position measured the distance with dynamic constriction factors *k*. In order to improve PSO's convergence and accuracy on complex problem, we present the ALPS utilizing a new learning strategy. This paper is organized as follows. Section 1 presents an overview of the PSO with Inertia Weight and PSO with Constriction Factors. Section 2 describes the Adaptive constriction factor for Location-related Particle Swarm. Section 3 introduces the test functions used to evaluate the ALPS and experiment results. Finally, conclusions and future research are given in Section 4.

2 ALPS Algorithm

The Adaptive constriction factor for Location-related Particle Swarm (ALPS) is introduced in this section. In ALPS, all particles are arranged in a dynamics that shift the constriction factor *k*. In the traditional, particle swarm with constriction factor which each particle with that settings is dependent on c_1, c_2 parameters. In that situation, the overall search time of the constriction factor is constant. However, most of real problem is very difficulties that have many local optima situation and wide search space. That has need huge amount of computation cost with operating time. Furthermore, those are arduous problem, e.g., local optima trap, which difficult effective avoid for current techniques.

The socio-cognitive learning approach defined in the standard PSO. The ALPS algorithm make a new regulate the velocity to this approach: Each particle are mutually compatible, that constriction factor vary from adaptive algorithm. We take our inspiration from the study group at school. It is vary simply concept of the tactic learning of the student life. We consult with ahead of students in order to attempt to strengthen our learning efficiency. We know the top person at school, although we give preference to emulation of better and some person for the most part. Usually, we need to consult with more then one person as maybe there are a lot of difficulties that we wants imitate top-one person.

Through simulation, we observed that particles are highly probable that these rapidly rush to a local optimum solution and stagnate due to the lack of momentum in the tradition of velocity approach. Indeed, tradition of velocity approach is highly influence on the global best position. That can be higher speed close to global best position, especially in the best solution found so far, but we are not sure whether this global best will optimal solution. Hence, we adopt distinct evaluation approach to put a bridle on velocity. It can be seen from above discussion that Contraction-Expansion Constriction Factors *k* influences the velocity manipulation of the individual particle, and therefore exerts significant influence on convergence of the PSO-cf algorithm. we deemed that it help particle reason the difference position between the better position and oneself. For such reason, we propose an estimate mean position in better (This import of whatever particles experienced over average performance) experiential space. The mathematical repesentation of this concept is given by (6). Such position is called core position.

$$core_d = \frac{\sum_{j=1}^{m} \hat{p}_{jd}}{m} \quad (6)$$

Where each particle *j* represents the over average performance of the previous position $\hat{p}_{jd}$, which determines core position. *m* is number of which over average performance of the previous position. The below expression is called the Location-related-Ratio, suggesting the name LR for the algorithm; those are derived from core position. In the LR method, those are identifying how close the particle is to the core position, $core_d$.

$$LR_{id} = \left|\frac{core_d - x_{id}}{core_d - MAX\left(x_{\cdot d}\right)}\right| \quad \left| \; x_{\cdot d} \in \left\{x_{1d} \ldots x_{nd}\right\} \right. \quad (7)$$

where $|\cdots|$ dentes the absolute value, and x_{id} is the position of the *i*th particle in the *d*th dimension. $MAX\left(x_{\cdot d}\right)$ represents the farthest position of the entire particle in the dimension *d*. $\left|core_d - x_{id}\right|$ is evaluated by the distance between the *i*th particle's current position and the core position in the *d*th dimension. If particle *i* is close to the core position, reduce the value of LR.

We make use of normalization method in order to adaptive parameter control with a wide range between κ_{max} and κ_{min}. The normalization method and new velocity vector formula were:

$$\kappa_{id} = \kappa_{\max} - LR_{id} \cdot (\kappa_{\max} - \kappa_{\min}) \quad (8)$$

$$v_{id} = \kappa_{id} \cdot \left(w \times v_{id} + c_1 \times \varphi_1 \times (p_{id} - x_{id}) + c_2 \times \varphi_2 \times (p_{gd} - x_{id}) \right) \quad (9)$$

Simulations were carried out with numerical benchmarks, to find out the best ranges of values for $\kappa_{\max}$ and $\kappa_{\min}$. An improved optimum solution for most of the benchmarks was observed when $\kappa_{\max}$ was set to 1 and $\kappa_{\min}$ was set to 0.1, over the full range of the search.

In summary, we proposed Adaptive constriction factor for Location-related Particle Swarm method for optimization problem can be described as follows:

- In swarm communication, an offer of distance detection while changes in unable to understand search space's trend for learning within the PSO.
- Flight performance, we adopt the Adaptive Algorithm to a great variety of constriction factors weighting. That mechanism affects the growth of capability of extend / shrink velocity vector.

Traditional particle swarm method, PSO use trajectories search of each individual experiences in the problem space, though which have according to flying experience of the other member due to current global best particle often is the best solution found so far by the swarm. For this reason, we propose a method; whole particles will find the elite's mean position and in which the optimization success relies the diversity of the method to not being trapped in a local optima. Therefore, we proposed ALPS algorithm is capable of locating a good solution at a significantly faster rate.

The pseudocode for **ALPS** is as follows.

define

$\hat{p}_{jd} \equiv (\hat{p}_{1d}, ..., \hat{p}_{md})$;

begin

initialize the population;

repeat:

for (*i*=1 to the population size)

evaluate the fitness:= $f(x_i)$ to update p_{id} and p_{gd};

$$w = (w_{\max} - w_{\min}) \times \frac{(MAXITER - iter)}{MAXITER} + w_{\min}$$

$$\hat{p} = \left(f(p_1) + ... + f(p_n) \right) / n ;$$

for (*d*=1 to the problem dimensionality)

if ($f(p_i) \le \hat{p}$)

$j \in i \cup \{j\}$;

increase *m*;

end-if

make the marker (core position):

$$core_d = \frac{\sum_{j=1}^{m} \hat{p}_{jd}}{m} ;$$

calculate the location-related ratio:

$$LR_{id} = \left| \frac{core_d - x_{id}}{core_d - MAX(x_{\cdot d})} \right| ;$$

normalize the constriction factor *k* :

$\kappa_{id} = \kappa_{\max} - LR_{id} \cdot (\kappa_{\max} - \kappa_{\min})$;

calculate the new velocity:

$$v_{id} = \kappa_{id} \times \left(w \times v_{id} + c_1 \times \varphi_1 \times (p_{id} - x_{id}) + c_2 \times \varphi_2 \times (p_{gd} - x_{id}) \right) ;$$

limit amplitude:

$v_{id} = \text{sign}(v_{id}) \cdot \min\left(\text{abs}(v_{id}, \text{V}_{\max}) \right)$;

update position:

$x_{id} = \text{sign}(x_{id}) \cdot \min\left(\text{abs}(x_{id}, \text{Max}_{\text{d}}) \right)$;

end-for *d*

end-for *i*

until stopping condition is true

end algorithm

Fig.1 Pseudocode for the ALPS algorithm.

3 Experiment Results

In this section, the experiments that have been done to compare the different algorithms for continuous function optimization are described. It is common to compare different algorithms using a large test set, especially when the test complicates function optimization, in the field of evolutionary computation. First, as we wish to test the ALPS on diverse benchmark functions and our main motive is to improve PSO's performance, we select unimodal function and multimodal benchmark as well as being functions that are robust enough to describe a wide selection of problem condition. As we wish to test the ALPS on optimization can achieve the effectiveness and efficiency results. The four of the will-known benchmarks used in evolutionary optimization metheds. Those were used to evaluate the performance, optimum solution after number of iterations and robustness after number of iterations. These benchmarks are widely used in evaluating performance of PSO methods, [1], [10], and [11]. The set of test functions (see Table 1) contains functions that are commonly used in the field of continuous function optimization. These functions are all minimization problems with minimum value zeros. Table 2 shows the values that have been used for dimension (variable) of these functions, range of search space, the initial range of the population also listed, that is asymmetry as used in [10], [11]. The "Criterion" column lists function value criterion which have to be achieved by the algorithm in the robustness analysis.

Table 1
Benchmarks for Simulations

Name of the function	Mathematical representation
Sphere function	$f_1(x)=\sum_{i=1}^{n} x_i^2$
Rosenbrock function	$f_2(x)=\sum_{i=1}^{n-1}\left[100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2\right]$
Ackley's function	$f_3(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}cos\left(2\pi x_i\right)\right)+20+e.$
Griewank function	$f_4(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}cos\left(\frac{x_i}{\sqrt{i}}\right)+1.$

Table 2
Parameters and Criteria for the Test Functions

Function	Dimension	Range of search	Range of initalization	Criterion
f_1	30	±100	$[50,\ 100]$	0.01
f_2	30	±2.048	$[1.024,\ 2.048]$	100
f_3	30	±30	$[-30,\ 16]$	5
f_4	30	±600	$[200,\ 600]$	0.1

In the four benchmark functions above, Ackley (f_3), Griewank (f_4) are multimodal functions while the others are unimodal. Furthermore, the unimodal functions are simple problems whereas the multimodal functions are designed with a considerable amount of local minima. The unimodal and multimodal functions as well as common and complex cases are included. Simulations were carried out to find the global minimum where they are located of some functions. The following functions have been used for evaluation of ALPS.

The experiments were conducted to compare seven PSO algorithms including the proposed ALPS algorithm on four test problems.

There were seven kinds of algorithm types:

- PSO with linearly decreasing inertia weight (PSO-w) [12];
- PSO with constriction factor (PSO-cf) [3]**;**
- PSO with time-varying acceleration coefficients (PSO-tc) [13];
- PSO with mutation (MPSO) [6];
- Cooperative PSO with split swarm (CPSO-s) [1];
- Self-Organizing Hierarchical Particle Swarm Optimizer (HPSO) [11];
- ALPS.

All experiments were run for 4×10^4 function evaluations(FEs). The number of iterations was chosen to correspond to 2×10^3 iterations of the PSO-w, PSO-cf, PSO-tc, MPSO, HPSO and ALPS (with population size of 20). All experiments were run 50 trials; the results reveal are averages best function and such 95% confidence interval calculated form all 50 trials. Table 3 shows the parameter values used in each PSO. In the experiments we chosen the parameter follow [1], [3], [11], [12], in which (see Table 3) we use "Δ" as the symbol for the parameter is linearly increasing weight was set to change i to f over the generations at position $[i, f]$, and then we use "∇" as the symbol for the parameter is linearly decreasing weight was set to change i to f over the generations at position $[i, f]$. Besides, we use "$\Diamond$" as the symbol for the adaptive parameter control with a wide range between h and l at position $[h,\ l]$. The "s" column lists the number of split swarm. The "p_m" column is the mutation probability. Last, we uses double-precision floating point of per variables to make the comparison between whole PSOs**.**

Table 3
Parameter values used in each PSO

Parameters /Algorithm	κ	w	s
PSO-w	—	$[0.9, 0.4]\nabla$	1
PSO-cf	0.7298	—	1
PSO-tc	—	$[0.9, 0.4]\nabla$	1
MPSO	—	$[0.9, 0.4]\nabla$	1
CPSO-s	—	$[0.9, 0.4]\nabla$	30
HPSO	—	$[0.9, 0.4]\nabla$	1
ALPS	$[1, 0.1]\Diamond$	$[0.9, 0.4]\nabla$	1

Parameters /Algorithm	c_1	c_2	p_m
PSO-w	2	2	—
PSO-cf	2.05	2.05	—
PSO-tc	$[2.5, 0.5]\nabla$	$[0.5, 2.5]\Delta$	—
MPSO	2	2	0.4
CPSO-s	2	2	—
HPSO	2	2	—
ALPS	2.05	2.05	—

Table 4 through 7 presents the experiment results. Those showing cover the average, confidence interval of 95% of the 50 runs and its significance of the seven algorithms on the four test problems. In light of results presented below, it is noteworthy that problem solving using the adaptive (extend / shrink) constriction factor for Location-related method have increased PSO work efficiency many times, relative to the other versions of PSO algorithms. For evaluating the significance the Mann-Whitney U test has been used to compare the results for two algorithms. The Mann-Whitney U test is equivalent to the Wilcoxon rank sum test. The tests performs a two-sided rank sum test of the hypothesis that two independent samples, in the two algorithms (x and y), come from distributions with equal medians, and returns the significance from the test, these results of the 50 trials. In the tests, there are compared using the null-hypothesis $H_0: x \geq y$ and the one-sided alternative hypothesis $H_a: x < y$ at a significance level of $\alpha = 0.01$. In the results the significance comparison amongs a set of seven algorithms is represented using the

7×7 matrix $M = m_{x,y} \big|_{x,y \in [1:7]}$, in which we use "X" as the symbol for the algorithm x is significantly better at position $m_{x,y}$. Besides, we use "—" as the symbol for the algorithm x is not significantly better than algorithm y at position $m_{x,y}$.

Table 4
Sphere (f_1) Function Evaluations

Algorithm		Mean	Mann-Whitney U test						
			1	2	3	4	5	6	7
PSO-w	(1)	2.74E+04 ± 3.50E+03		—	—	—	—	—	—
PSO-cf	(2)	2.72E+04 ± 3.12E+03	—		—	—	—	—	—
PSO-tc	(3)	5.20E+03 ± 1.99E+03	**X**	**X**		—	—	—	—
MPSO	(4)	1.21E-11 ± 6.35E-12	**X**	**X**	**X**		—	**X**	—
CPSO-s	(5)	9.20E-10 ± 1.37E-09	**X**	**X**	**X**	—		**X**	—
HPSO	(6)	5.60E+03 ± 1.90E+03	**X**	**X**	—	—	—		—
ALPS	(7)	**6.86E-25 ± 1.37E-24**	**X**	**X**	**X**	**X**	—	**X**	

Table 5
Rosenbrock (f_2) Function Evaluations

Algorithm		Mean	Mann-Whitney U test						
			1	2	3	4	5	6	7
PSO-w	(1)	4.60E+01 ± 3.42E+01		—	—	—	**X**	—	**X**
PSO-cf	(2)	6.83E+01 ± 6.56E+01	—		—	—	**X**	—	**X**
PSO-tc	(3)	9.85E+00 ± 7.71E+00	**X**	**X**		—	**X**	—	**X**
MPSO	(4)	**5.30E-01 ± 3.37E-01**	**X**	—	**X**		**X**	**X**	**X**
CPSO-s	(5)	4.71E+03 ± 2.65E+02	—	—	—	—		—	—
HPSO	(6)	3.42E+00 ± 3.35E+00	**X**	**X**	—	—	**X**		**X**
ALPS	(7)	6.12E+02 ± 2.24E+02	—	—	—	—	**X**	—	

Table 6
Ackley (f_3) Function Evaluations

Algorithm		Mean	Mann-Whitney U test						
			1	2	3	4	5	6	7
PSO-w	(1)	5.15E-01 ± 5.58E-01		**X**	**X**	—	—	**X**	—
PSO-cf	(2)	4.40E+00 ± 1.02E+00	—		—	—	—	—	—
PSO-tc	(3)	1.81E+00 ± 2.30E-01	—	**X**		—	—	—	—
MPSO	(4)	1.60E-06 ± 4.42E-07	**X**	**X**	**X**		**X**	**X**	—
CPSO-s	(5)	1.15E-04 ± 1.32E-05	—	**X**	**X**	—		**X**	—
HPSO	(6)	1.85E+00 ± 2.10E-01	—	**X**	—	—	—		—
ALPS	(7)	**1.20E-14 ± 1.19E-15**	**X**	**X**	**X**	**X**	**X**	**X**	

Table 7
Griewank (f_4) Function Evaluations

Algorithm		Mean	Mann-Whitney U test						
			1	2	3	4	5	6	7
PSO-w	(1)	1.97E+02 ± 3.06E+01		—	—	—	—	—	—
PSO-cf	(2)	1.39E+02 ± 2.99E+01	**X**		—	—	—	—	—
PSO-tc	(3)	1.98E+01 ± 1.18E+01	**X**	**X**		—	—	—	—
MPSO	(4)	1.25E-12 ± 6.89E-13	**X**	**X**	**X**		—	**X**	—
CPSO-s	(5)	1.87E-09 ± 2.32E-09	**X**	**X**	**X**	—		**X**	—
HPSO	(6)	1.09E+01 ± 9.79E+00	**X**	**X**	—	—	—		—
ALPS	(7)	**1.18E-16 ± 9.80E-18**	**X**	**X**	**X**	**X**	—	**X**	

Figures 2 through 5 present the results on the optimization functions defined in the previous section. The graphs show mean best results over 50 trials. The results show that the ALPS works better then other PSO algorithm except on Rosenbrock function. It is because that the ALPS has ability to evolve continuance, but that is search the solution space without converging in the iteration terminated. Rosenbrock function is a muti-dimensional function with a deep valley with the shape of a parabola. Due to the non-linearity of the valley, many algorithms converge slowly because they change the direction of the search repeatedly. It is a challenge for any optimization algorithm. Its difficulty is mainly due to the non-linear interaction among its variables. According to the "no free lunch theorem" (NFLT), It says that, "all algorithms that search for an minimum of a function perform exactly the same, when averaged over all possible functions."[14] ,and therefore Rosenbrock function is difficult to ward off the converge slowly for ALPS. Notwithstanding ALPS dose not perform the best for Rosenbrock function; we may not expect the best performance on all benchmark functions, as the ALPS focuses on improving the PSO's performance on great majority problems.

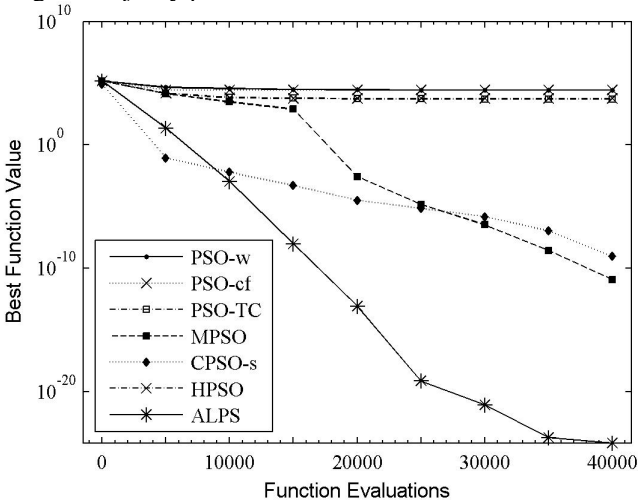

Fig.2 Sphere (f_1) mean best function value profile.

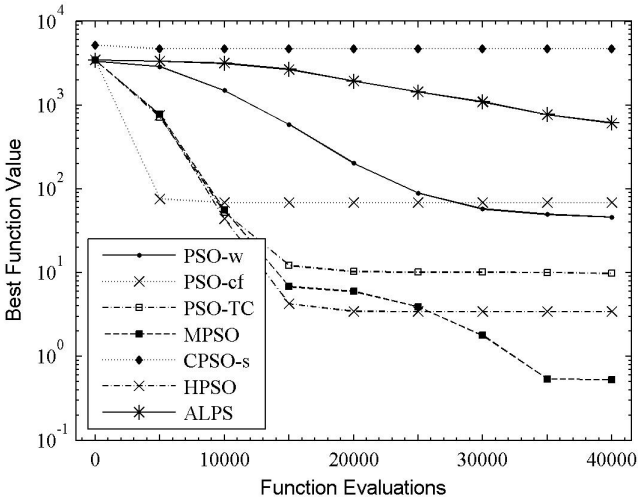

Fig.3 Rosenbrock (f_2) mean best function value profile.

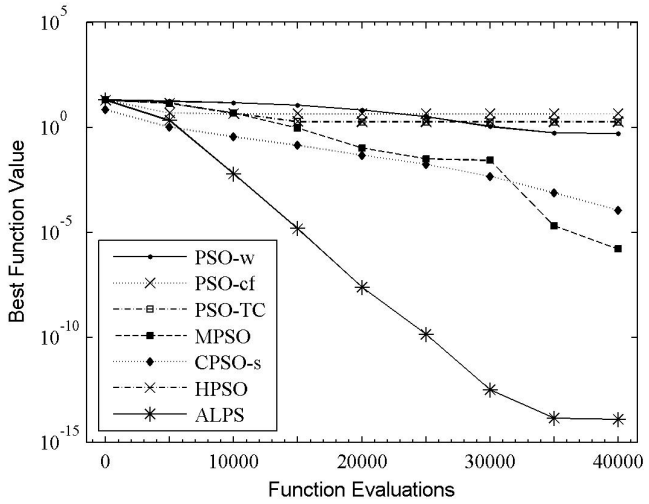

Fig.4 Ackley (f_3) mean best function value profile.

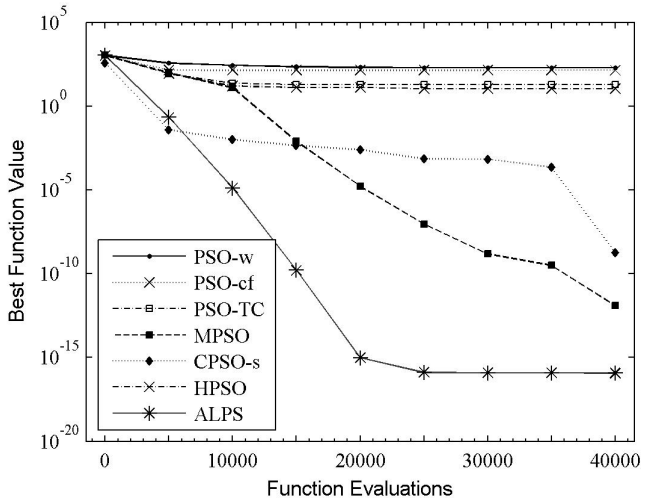

Fig.5 Griewank (f_4) mean best function value profile.

Table 8 shows the values that compares the various PSO algorithms to determine their relative rankings by both robustness and convergence speed as criteria. The criteria have defined in the Table 2. The "Robustness" is used here to mean that the algorithm succeeded in reducing function value below the criterion by the maximum number of function evaluations (FEs). The "Succeeded" column lists the number of trials (out of 50 trials) that managed to attain a function value below the criterion by 4×10^4 FEs, while the "Fn Evals" column record of the number of function evalutions needed on average to reach the criterion when the trial has succeeded. Overall, as far as robustiness is concerned the ALPS algorithm appears to be the winner. It performed a perfect score in three of the four benchmark function.

Table 8
Robustness Analysis

Algorithm	Sphere (f_1)		Rosenbrock (f_2)	
	Succeeded	Fn Evals.	Succeeded	Fn Evals.
PSO-w	2	34190	49	23935
PSO-cf	2	12120	46	2324
PSO-tc	25	16146	49	8831
MPSO	**50**	17512	**50**	8949
CPSO-s	**50**	4944	0	N/A
HPSO	25	16232	**50**	8825
ALPS	**50**	8790	16	28896
Algorithm	**Ackley (f_3)**		**Griewank (f_4)**	
	Succeeded	Fn Evals.	Succeeded	Fn Evals.
PSO-w	49	21810	5	27280
PSO-cf	39	3122	9	5753
PSO-tc	**50**	9809	40	11873
MPSO	**50**	9862	**50**	12333
CPSO-s	**50**	1200	**50**	3828
HPSO	**50**	9756	44	11862
ALPS	**50**	3336	**50**	5273

4 Conclusion

This paper proposes a novel technique to optimize problems using a continuous valued representation, adaptive approach, particles' dispersion position and distance method. The ALPS is compared to the PSOs. From the experimental results, it is clear that the ALPS generated desirable results. The main benefit of the ALPS representation the fact that the particle gets to capable of collaborate for the problems defined in this paper. The results represent, we proposes the ALPS method as a robust and consistent, that is able to local optimum elimination and better convergence rate for less population size and iteration. As experimentally, increasing the size of the population seems to improve the performance of the swarm. Nevertheless, population size, number of iteration (cost) with performance trade-off, which reduces the computational cost those take to lighten the load of the system. Mentionable, Compared with other PSO, the ALPS can be to look after both efficiency and the computational cost.

Moreover, the ALPS with a dynamically adjusting constriction factor of the velocity vector has been introduced, which could improve the performance of PSO with constriction factor (PSO-cf). Lastly attractive performance with characteristic of the ALPS is that it does not adopt any complex operations to the original PSO framework. The only difference from such is the velocity update equation, based on each particle's place. The ALPS is also simple and easy to implement like the PSO.

One interesting topic for further research, we plan to study more fully the effects of varying parameter values such as $\kappa_{\max}$ and $\kappa_{\min}$ on performance of the ALPS.

References:

[1] Van den Bergh, F. and Engelbrecht, A.P., A cooperative approach to particle swarm optimization, *IEEE Transactions on Evolutionary Computation*, Vol.8, 2004, pp.225-239.

[2] Clerc, M., The swarm and the queen: towards a deterministic and adaptive particle swarm

optimization, *Proceedings of the IEEE Congress on Evolutionary Computation*, 1999.

[3] Clerc, M., and Kennedy, J., The particle swarm-explosion, stability and convergence in a multidimensional complex space, *IEEE Transactions on Evolutionary Computation*, Vol.6, no.2, 2002, pp.58-73.

[4] Eberhart, R.C. and Shi, Y., Comparison between genetic algorithms and particle swam optimization, *Proceedings of the 7th Annual Conference on Evolutionary Programming*, Vol.1447, 1998, pp.611-616.

[5] Eberhart, R.C. and Shi, Y., Comparing inertia weights and constriction factors in particle swarm optimization, *Proceedings of the 2000 Congress on Evolutionary Computation*, Vol.1, La Jolla., 2000, pp.84-88.

[6] Higashi, N. and Iba, H., Particle swarm optimization with Gaussian mutation, *Proceedings of the IEEE Swarm Intelligence Symposium 2003 (SIS 2003)*, Indianapolis, IN, 2003, pp.72-79.

[7] Hu, X., Shi, Y. and Eberhart, R., Recent advances in particle swarm, *Proceedings of Congress Evolutionary Computation*, Vol.1, 2004, pp.90-97.

[8] Janson, S. and Middendorf, M., A hierarchical particle swarm optimizer and its adaptive variant, *IEEE Transactions on System, Man, and Cybernetics-Part B: Cybernetics*, Vol.35, No.6, 2005, pp.1272-1282.

[9] Kennedy, J. and Eberhart, R.C., Particle swarm optimization, *Proceedings of the Fourth IEEE International Conference on Neural Networks*. Perth, Australia, 1995, pp.1942-1948.

[10] Liang, J.J., Qin, A.K., Suganthan, P.N., and Baskar, S., Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE *Transactions on Evolutionary Computation*, vol.10, No.3, 2006, pp.281-295.

[11] Ratnaweera, A., Halgamuge, S.K., and Watson, H.C., Self-organizing hierarchical particle swarm optimizer with time varying accelerating coefficients, *IEEE Transactions on Evolutionary Computation*, vol. 8, 2004, pp. 240–255.

[12] Shi, Y. and Eberhart, R.C., A modified particle swarm optimizer, *Proceedings of IEEE World Congress Computation Intelligence*, 1998, pp.69–73.

[13] Suganthan, P. N., Particle swarm optimizer with neighborhood operator, *Proceedings of IEEE International Congress Evolutionary Computation*, Vol.3, 1999, pp.1958–1962.

[14] Wolpert, D.H. and Macready, W.G., No free lunch theorems for optimization, *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 1, April 1997, pp.67-82.