A New Cooperative PSO Approach for the Optimization of Multimodal Functions

Ruei-Yang Wang   Wei-Po Lee   Yu-Ting Hsiao

Abstract—**Particle swarm optimization (PSO) has been proposed to solve many optimization with good results. However, more efficient strategies are still needed to control the trade-off between exploitation and exploration in the search process for solving complex tasks. In this work, we propose a new PSO approach to overcome the search difficulties. Our approach focuses on two search strategies for multimodal functions. One is a cooperative strategy that controls search region and integrates partial and full dimension PSO search. The other is to control the velocity of the particles in an adaptive way, according to how they move in the space. Experiments have been conducted to evaluate the proposed approach, and the results show that our approach can perform better than other popular PSO variants.**

***Index Terms*—swarm intelligence; particle swarm optimization; cooperative strategy; multimodal function optimization**

I. Introduction

Particle swarm optimization (PSO, [1]) has been proposed as an alternative to traditional evolutionary algorithms (EAs). It attempts to mimic the goal-seeking behavior of biological swarms. In PSO, a possible solution of the optimization problems is represented as a particle and the algorithm operates in an iterative manner. Unlike traditional EAs, particles in PSO do not perform the operator of genetic recombination between particles, but they work individually with social behavior in swarms. PSO has some attractive characteristics. In particular, it has memories, so that knowledge of good solutions can be retained by all particles (solutions). This method has been successfully used to solve many discontinuous and complex problems with good results [2][3].

As can be observed, when an application task involves many parameters and the parameter dimension increases to match the increase in task complexity, the solution space grows exponentially. Consequently, the search becomes more and more difficult. Though different modified PSO algorithms have been proposed to give better solutions than the standard PSO does, their search quality declines soon for complex tasks with high dimensional and multimodal objective functions. In addition, as the distribution and density of these optimal solutions vary from function to function, it is difficult to design a general or universal strategy for all problem situations. This is mainly because of that PSO has a high convergence speed and this often results in the loss of diversity during the optimization process. The undesirable premature situation causes particles to get trapped in local optimums and unable to gain the best solution. Therefore, an efficient strategy to well-control the trade-off between exploitation and exploration in the search process is still under investigation.

To overcome the search difficulties described above, in this work we propose a new PSO approach with two special features: dimension partition and adaptive velocity control. Dimension partition is to control search direction and region. It involves a cooperative strategy that concurrently exploits the advantages of both single dimension and full dimension PSO algorithms. And adaptive velocity control means to dynamically regulate the flying speed of the particles, according to how they move in the space. To verify the proposed approach, extensive experimental runs have been conducted and comparisons between our approach and some famous PSO variants have been made. The results confirm the performance of our approach. They show that the proposed approach performs the best in most of the test functions, especially when the test functions are rotated to increase the complexity.

II. Background

Similar to other EAs, PSO is also a population-based technique. The basic PSO algorithm contains a set of particles and operates in an iterative manner. Each particle is characterized by its position and velocity, which moves in the search space. The position of each particle represents the potential solution and is evaluated by a predefined evaluation (fitness) function. During the iterative search process, each particle remembers its previous best position and the best position of any particle in the swarm. Then, the particle uses the above position information to modify its position and velocity, and continues its movement in the search space.

To enhance the performance of individuals, the main operator in the PSO is velocity updating for the particles, which combines the best position obtained by the swarm of particles and the best position reached by a certain particle during its flying history. It has the effect that the particles move towards the best position of the swarm. In the original PSO, the velocity and the position of a particle at time step t+1 are updated from those at time step t by the following rules:

W.-P. Lee, R.-Y. Wang and Y.-T. Hsiao are with the Department of Information Management, National Sun Yat-sen University, Kaohsiung, Taiwan (corresponding author's e-mail: wplee@ mail.nsysu.edu.tw).

$$v_{id}^{t+1} = v_{id}^{t} + c_1 r_1^t (p_{id}^t - x_{id}^t) + c_2 r_2^t (p_{gd}^t - x_{id}^t)$$

$$x_{id}^{t+1} = x_{id}^{n} + v_{id}^{t+1}$$

In the above equations, v_{id} and x_{id} are the velocity and position of particle i at dimension d ($1 \leq d \leq n$, n is the dimensionality of the search space) and p_{id} is the previous best position of particle i, while p_{gd} is the best position of the swarm. The coefficients c_1 and c_2 are two positive acceleration constants used to scale the contribution of the cognitive and social components; they are often determined empirically. In addition, r_1 and r_2 are random values within the range [0, 1]. The products c_1r_1 and c_2r_2 thus stochastically control the overall velocity of a particle. With a newly obtained velocity, the position of a particle is then updated accordingly. In addition, a maximal flying speed v_{max} is often used to restrict the flying of the particles.

To balance the effect between local and global search, Shi and Eberhart proposed to use a weighting factor with value between 0 and 1 in the velocity rule [4], which was then modified as:

$$v_{id}^{t+1} = wv_{id}^{t} + c_1 r_1^t (p_{id}^t - x_{id}^t) + c_2 r_2^t (p_{gd}^t - x_{id}^t)$$

In the above equation, w is the inertia weight, which controls the momentum of the particle by weighting the contribution of the particle's previous velocity (i.e., the influence of the memory). In addition, Clerc and Kennedy introduced a constriction factor to the above equation to further control the flight of the particles [5]. It is to constrict the velocities of particles and achieve the exploration-exploitation balance during the search and to guarantee that the particles converge to a stable point. The velocity updating rule then became as the following in which χ is the constriction factor and φ is a parameter often used to control the convergence characteristics of the PSO:

$$v_{id}^{t+1} = \chi(wv_{id}^{t} + c_1 r_1^t (p_{id}^t - x_{id}^t) + c_2 r_2^t (p_{gd}^t - x_{id}^t))$$

$$\chi = 2 / \left| 2 - \phi - \sqrt{\phi^2 - 4\phi} \right| \quad \text{where } \varphi = c_1 + c_2, \varphi > 4$$

As can be observed, the above velocity control rule can be easily modified to fulfill the specific requirements of different applications, and there have been a large amount of PSO variants proposed to solve the relevant problems. Without losing generality, in the following we will take the perspectives of dimension and landscape of solution space to analyze several most efficient and relevant PSO methods. One important and most relevant variant is the cooperative PSO (CPSO, [6]) method that performed dimension partition of solution space to reduce the search complexity. CPSO originated from the cooperative coevolutionary genetic algorithm (CCGA, [7]), in which partial solutions were derived from the decomposed sub-spaces and then combined together to form the complete solution. CPSO includes two versions, CPSO-S and CPSO-H. CPSO-S is a direct application of Potter's CCGA on the original PSO, while in the enhanced version, CPSO-H, the authors alternately performed the one dimensional search and the original PSO (full dimensions) search to obtain better results.

The other type of variants is to take into account the particle performance and the integrated effect of particles in the swarm, rather than to consider only the effect of best particle. For example, the fully informed PSO (FIPS, [8]) took the way of weighted sum of all particles to update velocity of a certain particle in the rule described above. In fitness-distance ratio PSO (FDR-PSO, [9]), the authors chose to update one particle for each dimension to overcome the counteraction effect in FIPS. Also, Liang *et. al* presented a new method called comprehensive learning PSO (CLPSO, [10]) that randomly selected a particle as the target model rather than to select the best particle in the swarm. The experimental results have shown that this method and FIPS can only solve multimodal problems, rather than unimodal problems. In addition, some PSO variants developed performance-based strategies to re-organize particles in the same swarm to try and alleviate premature convergence, for example adaptive PSO (APSO, [11]) and efficient population utilization strategy PSO (EPUS, [12]).

III. The Proposed Approach

To enhance the PSO performance, we present a new approach for the optimization of multimodal functions. Our approach includes two major steps. The first step is to deal with high dimensional multimodal problems through a two-swarm cooperative strategy that improves the original PSO search from different viewpoints. And in the second step, an adaptive strategy is developed to dynamically control the particle velocity during the optimization procedure. The details are described in the following.

A. *Dimension Partition*

As mentioned above, the search complexity increases dramatically when the search space grows exponentially along with the increase of problem dimension. Especially when the dimensions are not independent (they rely on one another), the search becomes even more difficult. One promising way to tackle this scalability problem is to exploit the principle of cooperative search, which is to separately search different dimensions (regions) and then integrate the results. Therefore, inspired by the CCGA and CPSO methods, we develop a new two-swarm PSO to deal with high dimensional multimodal functions and their variants (rotated or/and shifted functions).

In our PSO approach, the swarm is divided into two parts: group-A and group-B. The original PSO computation is applied to group-A to conduct full dimensional global search. In addition, a single dimension PSO is developed and performed on group-B, in which the particles search only one dimension at a time. The dimension to be searched starts from the first one, and then takes turn in a cyclic manner. A communication phase is also designed for information sharing (i.e., to exchange the best results of the two groups). Different from the CPSO method in which the algorithms (one full dimension PSO and one single dimension PSO) for the two swarms perform the search sequentially, the two PSO groups in our approach work concurrently. Most importantly, in our approach, group-A and group-B continuously operate for a certain number of iterations before the communication happens, whereas in CPSO, the two swarms exchange results very frequently (i.e., after each iteration). It should be noted that the frequent communication often causes the premature convergence: situation in which particles are trapped to local minima, and this problem is especially serious in the functions with interdependent dimensions (variables). According to our design, during the search period before the communication phase, the single dimension PSO for group-B searches the same dimension so that the solutions can be improved progressively and stably.

Fig. 1 illustrates the flow. As can be seen, when the communication occurs, group-A duplicates the best solution (i.e., *best-A*, represented as a vector $(x_1, x_2, ..., x_n)$) obtained so far and sends it to group-B. In each search interval (i.e., k iterations, and k is 50 in our current implementation, determined empirically), the particles in group-B take the best solution obtained from group-A as the *context vector* (which is required to provide a suitable context so that each particle in group-B can be evaluated). Then these particles search for a most appropriate value x_d^{best} in the current dimension d $(1 \leq d \leq n)$ to form a complete solution $(x_1, x_2, ..., x_d^{best}, .., x_n)$. All dimensions will be searched one by one in turn and in a cyclic manner. That is, for a n-dimensional problem, the single dimension PSO procedure needs to continue $(k \times n)$ iterations to complete a search round for all dimensions.

Similarly, the single dimension PSO procedure provides the best result (derived for the current dimension) of group-B (i.e., *best-B*) to group-A. Then the full dimension PSO procedure tries to use *best-B* to substitute the previous best result of each particle P_i (called $best_i$) in group-A during its flying history, depending on whether such a substitution can lead to a better solution. If there is any performance improvement observed, the single-dimension PSO for group-B will start the new round and continue the search along the next dimension. On the contrary, once none of the particle values in group-A can be replaced (meaning that the PSO procedure for group-A has reached an optimal solution, local or global), the single dimension PSO procedure stops to operate and the particles in group-B are re-organized. Group-B is divided into two equal parts: one part (including half of the particles) merged to group-A to perform local search around the optimal solution found; and the other part, replaced by random particles to maintain swarm diversity and then the newly initialized particles perform full dimension PSO as those in group-A. With the dimension partition strategy described above, our PSO approach can deal with high dimensional problems and obtain good results.

B. *Adaptive Velocity Control*

In addition to the partition of search dimensions for performing cooperative search, we develop a new adaptive strategy to control the flying velocity of the particles. As is well-known, in solving optimization problems, to obtain high quality solutions (i.e., efficiency) in limited time (i.e., effectiveness), an iterative search-based approach needs to work with a well-designed strategy to obtain balance between search exploitation and exploration. In a PSO-based method, controlling particle velocity can achieve such a regulating effect, and the parameter v_{max} described in Section 2 plays an important role. With a large v_{max}, a PSO method can better explore the complete solution space; on the contrary, a small v_{max} directs the method to perform a local search. As a result, a PSO method tends to adopt a higher v_{max} value in the early stage of the search process, and a lower value later. A simple but popular way is to use a decreasing function (linear or others) to gradually reduce the v_{max} value (e.g., in proportion to the iteration number), and the corresponding effect has been confirmed [13][14]. However, a better control strategy is still needed for further performance enhancement.

In our work, the control strategy is to change the v_{max} value dynamically, depending on how the particles move in the space. A simple rule is designed to reason the motion status of each

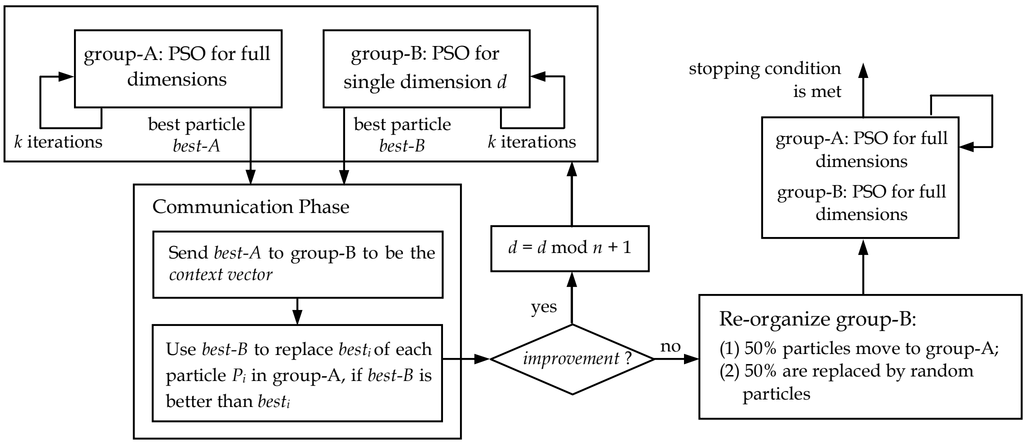
group-A: PSO for full dimensions
group-B: PSO for single dimension d
k iterations
best particle best-A
best particle best-B
k iterations
Communication Phase
Send best-A to group-B to be the context vector
Use best-B to replace best_i of each particle P_i in group-A, if best-B is better than best_i
d = d mod n + 1
yes
improvement ?
no
stopping condition is met
group-A: PSO for full dimensions
group-B: PSO for full dimensions
Re-organize group-B:
(1) 50% particles move to group-A;
(2) 50% are replaced by random particles

Figure 1. The main flow of the proposed dimension partition strategy.

particle, according to how it moves in two consecutive steps. The rule is to speculate if a particle has flied too fast so as to miss an optimal position. This occurs when the particle moved to a better position (i.e., the performance has been improved) at the last step and then to a worse position at the current step. Once the motion statuses of all particles have been obtained, information from all particles is assembled for controlling the swarm's behavior. This means when most of the particles (i.e., more than half of the particles in the swarm) show the over-flying situation (i.e., moving too fast), the v_{max} value is decreased to slow down the swarm to perform elaborate (local) search. The speed decreasing is estimated by the following equation:

$$v_{max}^{t+1} = \alpha \times v_{max}^{t}$$

Here, α is the rate to slow down the swarm. For unimodal functions that have relatively simple landscapes than multi-modal functions, a small value α is more suitable (with a manifest effect for slowing down the swarm). On the contrary, a large value is expected for multimodal functions. In this work, a value of 0.9 is used (determined by the preliminary tests).

To avoid the situation of search stagnation caused by the monotonic speed reduction, our dynamic strategy also allows the particles to not only slow down, but also speed up again. This is achieved by two rules. The first rule is used in the later stage of the search process, when all particles have moved for a number of iterations. It is to observe the moving variations (called Δd_i for dimension i, $1 \le i \le n$) of each particle along all dimensions, at the time of updating the best position of the swarm (i.e., p_{gd}). If the current v_{max} value is smaller than half (determined empirically) of the maximal variation (i.e., $v_{max} < 0.5\times\Delta d_{max}$, $\Delta d_{max} = \max(\Delta d_1, \Delta d_2, \ldots, \Delta d_n)$) recorded from certain dimension, meaning that the particle needs to move more than twice with highest speed in order to reach the global best position, we can then assume that v_{max} now has a relatively low value. In this situation, the rule will scale v_{max} up to half of the maximal variation mentioned above ($0.5\times\Delta d_{max}$). One the contrary, the second rule suits for all search iterations. In fact, it is not to increase the current v_{max} value, but allows all particles to exceed the v_{max} toward a predefined upper limit with a random probability. In our current implementation, the upper limit is 20 times of the original v_{max} value, and the probability is 0.2 for the multimodal functions here. In addition, a minimal velocity v_{min} is used in our velocity control strategy to ensure that the particles can move with a reasonable speed. Here, the v_{min} is defined to be the one tenth of the v_{max}.

IV. Experiments And Results

To verify the proposed approach, we conducted two sets of experiments to compare our approach with two famous PSO variants, APSO [11] and EPUS [12], which were recently published and have been shown to deliver better performance than most of the other PSO algorithms. Because the implementation details of these algorithms are not available, in order to keep the comparisons as objective as possible, we chose to collect results (including the results obtained by the above algorithms and those listed in their reports for performance verification) directly from their original research reports but not to re-implement the algorithms. To examine the generality of our approach, in each set of experiments we have used the same settings as the algorithms mentioned above, though which were best-fitted to the individual work but may not be particularly suitable for ours.

In the two sets of experiments, extensive runs have been carried out with several multimodal test functions collected from the literature [11][12]. The functions are summarized in the Table I. These functions have different aspects and characteristics. For example, Ackley function (f_6) has one narrow global optimum basin and many minor local optima, while Schwefel function (f_1) has deep local optima that are far from the global optimum. In the multimodal functions used, except Rastrigin function (f_5) and Ackley function (f_6), the variables are interdependent so that they become difficult to solve by simple algorithms (such as relaxation method). To ensure that there was sufficient correlation between the variables and to make the functions even harder to solve, some functions were further rotated from their original forms. The dimension for each function was 30.

Table I. The multimodal test functions.

f_1: Schwefel (original)	f_5: Rastrigin (original, rotated)
f_2: Generalized Penalized 1 (original)	f_6: Ackley (original, rotated)
f_3: Generalized Penalized 2 (original, rotated)	f_7: Griewank (original, rotated)
f_4: Noncontinuous Rastrigin (original, rotated)	f_8: Weierstrass (original, rotated)

Special calculations have been conducted to obtain the evaluation results of the rotated functions. To derive the result of a parameter vector x in the rotated function of an original function f, we took the commonly used method indicated in [15] to generate an orthogonal matrix M for x, multiply M and x to obtain a new parameter vector y (i.e., $y = M\times x$), and then calculate $f(y)$ as the result. In the experiments, a new orthogonal matrix was generated for each evaluation, in order to obtain a more objective performance evaluation. Because each new parameter y_i in the vector y was the linear combination of original parameters (i.e., $x_1 \sim x_n$ in the vector x), the functions that can be partitioned and directly solved by performing one dimension search n times became unsolvable. It thus provided more extensive performance verification for different algorithms. The test functions and the types of the functions are indicated in the appendix.

A. Comparing with APSO

The first set of experiments was to compare our approach with the APSO algorithm (Adaptive Particle Swarm Optimization, [11]) which evaluated the particle states (exploration, exploitation, convergence or jumping out) during the optimization process and decided the parameter settings

Table II. The mean and standard deviation of the runs for each test function. All functions are the original type.

Function		GPSO	LPSO	VPSO	FIPS	HPSO-TVAC	DMS-PSO	CLPSO	APSO	this work
f_1	*mean*	-10090.16	-9628.35	-9845.27	-10113.80	-10868.57	-9593.33	-12557.65	**-12569.50**	-12557.64
	std. dev	495.00	456.54	588.87	889.58	289.00	441.00	36.20	0.00	35.53
f_2	*mean*	1.04E-02	2.18E-30	3.46E-03	1.22E-31	7.07E-30	2.05E-32	1.59E-21	3.76E-31	**1.57E-32**
	std. dev	3.16E-02	5.14E-30	1.89E-02	4.85E-32	4.05E-30	8.12E-33	1.93E-21	1.20E-30	4.61E-40
f_4	*mean*	1.55E+01	3.04E+01	2.13E+01	3.59E+01	1.83E+00	3.28E+01	1.67E-01	4.14E-16	**1.95E-141**
	std. dev	7.40E+00	9.23E+00	9.46E+00	9.49E+00	2.65E+00	6.49E+00	3.79E-01	1.45E-15	1.05E-140
f_5	*mean*	3.07E+01	3.49E+01	3.41E+01	3.00E+01	2.39E+00	2.81E+01	2.57E-11	5.80E-15	**4.01E-116**
	std. dev	8.68E+00	7.25E+00	8.07E+00	1.09E+01	3.71E+00	6.42E+00	6.64E-11	1.01E-14	2.16E-115
f_6	*mean*	1.15E-14	1.85E-14	1.40E-14	7.69E-15	2.06E-10	8.52E-15	2.01E-12	1.11E-14	**6.84E-15**
	std. dev	2.27E-15	4.80E-15	3.48E-15	9.33E-16	9.45E-10	1.79E-15	9.22E-13	3.55E-15	6.37E-16
f_7	*mean*	2.37E-02	1.10E-02	1.31E-02	**9.04E-04**	1.07E-02	1.31E-02	6.45E-13	1.67E-02	2.29E-03
	std. dev	2.57E-02	1.60E-02	1.35E-02	2.78E-03	1.14E-02	1.73E-02	2.07E-13	2.41E-02	4.58E-03

(such as inertia weight, acceleration coefficients) accordingly. Table II lists the functions and the search settings were the same as in [11]. In the experiments, the number of particles was 20 and the number of evaluations was 200,000. For each function, 30 independent runs were conducted. The mean and standard deviation are presented in Table I (arranged as in [11]).

In Table II, the results of the first seven PSO algorithms (including GPSO [4], LPSO [16], VPSO [17], FIPS [8], HPSO-TVAC [18], DMS-PSO [19], and CLPSO [10]) that have been used in [12] for performance verification are also listed here, and the values are taken directly from the APSO article. The results of our approach are presented in the right-most column. To compare different algorithms, the best results (i.e., means) for each test function are marked in bold for identification. As can be seen, the proposed approach can give the best results in 4 (i.e., f_2, f_4, f_5, f_6) out of the 6 test functions, and APSO, 1 (i.e., f_1). In addition, the small deviations obtained from the experiments for all test functions also indicate the stability and repeatability of our approach. These results show that our approach is very promising for the optimization of multimodal functions.

B. *Comparing with EPUS*

The second set of experiments was to compare the proposed approach with the EPUS algorithm (Efficient Population Utilization Strategy for PSO, [12]). To conduct an efficient search, this algorithm created a swarm manager to include new particles and to remove some invalid particles. It also developed two information sharing strategies to prevent particles from falling into the local minimum. According to [12], the EPUS algorithm has been compared to the standard PSO ([20]) and three important PSO variants, including CPSO [6], CLPSO [10], and UPSO [21], on several test functions, and the results showed that EPUS performed better than other methods.

In this set of experimental comparisons, the 10 multimodal test functions, including 5 original and 5 rotated functions, and experimental settings presented in [12] were used. The number of particles was 20 and the number of evaluations was 150,000. For each test function, 25 experimental runs were conducted. As presented in [12], the mean and standard deviation for each function are listed in Table III and different algorithms are compared (as in [12]). This table shows that for the original test functions, EPUS performed better on f_3, and f_6, and our approach was better on f_8. Though EPUS seems to have better results than the proposed approach also on f_4 and f_5, the zero standard deviations reveal that the tiny performance difference of the two approaches on the two functions is likely to be caused by the value precision rather than the algorithms. As for the rotated functions, our approach obviously outperformed EPUS (and others). The proposed algorithm gave the best results on 3 (f_5, f_6 and f_8), and EPUS, 2 (f_3 and f_4) out of the 5 rotated functions. These results indicate the effectiveness and superior performance of the proposed approach.

V. CONCLUSIONS

Traditional PSO algorithm has a high convergence speed that often results in the loss of diversity during the optimization process. It needs to be improved in order to solve tasks with high dimensional multimodal objective functions. In this work, we propose a new cooperative PSO approach to overcome the search difficulties. For multimodal functions, our approach performs dimension partition that involves a cooperative strategy to integrate partial and full dimension search techniques. The hybrid search strategy can exploit the merits of both techniques to solve the scalability problem and find solutions for functions with large amount of interdependent variables. In addition, an adaptive velocity control strategy is developed. It calculates the motion status of each particle to dynamically speed up and slow down the particles. To evaluate the proposed approach, two sets of experimental comparisons have been conducted. The experimental comparisons were carried out in a peer-to-peer manner, and in each set of experiments, the PSO parameters were set to fit in the algorithms to be compared. The preliminary results show that our approach still outperformed others in most of the test cases.

Table III. The mean and standard deviation of the runs for each test function.

Type	Function		CLPSO	CPSO	UPSO	PSO-2006	EPUS	this work
original	f_3	*mean*	1.62E+03	2.58E+08	1.34E+01	2.91E+01	**1.35E-32**	1.46E-32
		std. dev	5.92E+03	4.41E+08	2.34E+01	2.93E+01	5.47E-48	3.20E-33
	f_4	*mean*	2.02E-07	8.84E-11	8.40E+01	1.03E+02	**0.00E+00**	**9.77E-65**
		std. dev	2.35E-07	2.21E-10	1.62E+01	4.16E+01	0.00E+00	4.78E-64
	f_5	*mean*	1.87E-09	3.15E-10	7.63E+01	7.73E+01	**0.00E+00**	**2.69E-73**
		std. dev	5.34E-09	1.05E-09	1.45E+01	1.93E+01	0.00E+00	1.32E-72
	f_6	*mean*	9.90E-14	3.90E-06	1.33E+00	7.77E+00	**3.91E-15**	6.96E-15
		std. dev	3.80E-14	5.98E-06	8.58E-01	2.09E+00	1.07E-15	0.00E+00
	f_8	*mean*	2.06E+00	4.12E+00	1.94E+00	8.97E-01	3.21E-02	**7.46E-05**
		std. dev	4.80E-01	1.72E+00	1.13E+00	8.40E-01	8.11E-02	1.69E-04
rotated	f_3	*mean*	4.36E-01	1.09E+06	2.77E+00	6.98E+00	**1.35E-32**	3.46E+01
		Std. dev	5.55E-01	2.37E+06	6.22E+00	8.65E+00	5.47E-48	5.77E+01
	f_4	*mean*	4.05E+01	8.38E+01	9.09E+01	1.06E+02	**3.03E+01**	3.73E+01
		std. dev	7.91E+00	2.68E+01	1.94E+01	2.84E+01	1.11E+01	1.70E+01
	f_5	*mean*	4.70E+01	9.52E+01	7.70E+01	8.17E+01	3.76E+01	**3.60E+01**
		std. dev	6.85E+00	2.93E+01	1.70E+01	2.09E+01	1.48E+01	1.17E+01
	f_6	*mean*	2.22E-06	1.52E+00	1.63E+00	6.76E+00	6.81E-01	**6.30E-07**
		std. dev	8.29E-06	8.30E-01	9.49E-01	2.28E+00	1.02E+00	2.27E-06
	f_8	*mean*	1.51E+01	1.36E+01	2.02E+01	1.72E+01	4.26E+00	**1.95E+00**
		std. dev	1.93E+00	3.56E+00	4.19E+00	3.52E+00	2.97E+00	1.52E+00

References

[1] J. Kennedy and R. Eberhart, Swarm Intelligence. CA: Morgan Kaufmann Publishers, 2001.

[2] F. Grimaccia, M. Mussetta, and R. Zich, “Genetical swarm optimization: self-adaptive hybrid evolutionary algorithm for electromagnetics,” IEEE Trans. on Antennas Propagation, 55, pp. 781–785, 2006..

[3] R. Poli, “Analysis of the publications on the applications of particle swarm optimisation,” Journal of Artificial Evolution and Applications, pp. 1-10, DOI:10.1155/2008/ 685175, 2008.

[4] Y. Shi and R. C. Eberhart, “A modified particle swarm optimizer,” Proceedings of the IEEE International Conference on Evolutionary Computation, pp. 69-73, 1998.

[5] M. Clerc and J. Kennedy, “The particle swarm-explosion, stability, and convergence in a multidimensional complex space,” IEEE Trans. on Evolutionary Computation, 6, pp. 58-73, 2002.

[6] F. van den Bergh and A. P. Engelbrecht, “A cooperative approach to particle swarm optimization,” IEEE Trans. on Evolutionary Computation, 8, pp. 225-239, 2004.

[7] M. A. Potter and K. A. de Jong, “A cooperative coevolutionary approach to function optimization,” Parallel Problem Solving from Nature, Vol.3, pp. 249-257, 1994.

[8] R. Mendes, J. Kennedy, and J. Neves, “The fully informed particle swarm: simpler, maybe better,” IEEE Trans. on Evolutionary Computation, 8, pp. 204-210, 2004.

[9] T. Peram, K. Veeramachaneni, and C. K. Mohan, “Fitness-distance-ratio based particle swarm pptimization,” Proceedings of IEEE Swarm Intelligence Symposium, pp. 174-181, 2003.

[10] J.J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, “Comprehensive learning particle swarm optimizer for global optimization of multimodal functions,” IEEE Trans. on Evolutionary Computation, 10, pp. 281-296, 2006.

[11] Z. H. Zhan, J. Zhang, Y. Li, and H. Chung, “Adaptive particle swarm optimization,” IEEE Trans. on Systems, Man, and Cybernetics-Part B, 39, pp. 1362-1381, 2009.

[12] S. T. Hsieh, T. Y. Sun, and C. C. Liu, “Efficient population utilization strategy for Particle swarm pptimizer,” IEEE Trans. on Systems, Man, and Cybernetics-Part B, 39, pp. 444-456, 2009.

[13] J. F. Schutte and A. A. Groenwold, “Sizing design of truss structures using particle swarms,” Structural and Multidisciplinary Optimization, 25, pp. 261-269, 2003.

[14] H. Y. Fan, “A modification to particle swarm optimization algorithm,” Engineering Computations, 19, pp. 970-989, 2002.

[15] R. Salomon, “Reevaluating genetic algorithm performance under coordinate rotation of benchmark functions,” BioSystems, 39, pp. 263-278, 1996.

[16] J. Kennedy and R. Mendes, “Population structure and particle swarm performance,” Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1671-1676, 2002.

[17] A. Ratnaweera, S. Halgamuge, and H. Watson, “Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients,” IEEE Trans. Evolutionary Computation, 8, pp. 240-255, 2004.

[18] J. J. Liang and P. N. Suganthan, “Dynamic multi-swarm particle swarm optimizer with local search,” Proceedings of IEEE Congress Evolutionary Computation, pp. 522-528, 2005.

[19] http://www.particleswarm.info/Standard_PSO_2006.c

[20] K. E. Parsopoulos and M. N. Vrahatis, “UPSO—A unified particle swarm optimization scheme,” Lecture Series on Computer and Computational Sciences, Vol. 1, Proceedings of the International Conference of Computational Methods in Sciences and Engineering, pp. 868-873, 2004.

[21] A. Renato Krohling and S. C. Leandro, “PSO-E: particle swarm with exponential distribution,” Proceedings of IEEE Congress on Evolutionary Computation, pp. 1428-1433, 2006.