2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

A Fast Restarting Particle Swarm Optimizer

Junqi Zhang†, Xiong Zhu, Wei Wang and Jing Yao
Department of Computer Science and Technology
Key Laboratory of Embedded System and Service Computing, Ministry of Education
Tongji University, Shanghai, 200092, China
† Corresponding author (*E-mail*: zhangjunqi@tongji.edu.cn)

***Abstract*—Particle swarm optimization (PSO) is a swarm intelligence technique that optimizes a problem by iterative exploration and exploitation in the search space. However, PSO cannot achieve the preservation of population diversity on solving multimodal optimization problems, and once the swarm falls into local convergence, it cannot jump out of the local trap. In order to solve this problem, this paper presents a fast restarting particle swarm optimization (FRPSO), which uses a novel restarting strategy based on a discrete finite-time particle swarm optimization (DFPSO). Taking advantage of frequently speeding up the swarm to converge along with a greater exploitation capability and then jumping out of the trap, this algorithm can preserve population diversity and provide a superior solution. The experiment performs on twenty-five benchmark functions which consists of single-model, multimodal and hybrid composition problems, the experimental result demonstrates that the performance of the proposed FRPSO algorithm is better than the other three representatives of the advanced PSO algorithm on most of these functions.**

I. Introduction

Particle swarm optimization (PSO) is an evolutionary computation technique which is firstly put forward by Eberhart and Kennedy in 1995 [1]. This algorithm was initially enlightened by the regular pattern of birds cluster activities and then established a simplified model that uses the swarm intelligent technology. Through sharing the information of particles, which exist in the group, the whole population moves from disorder to order evolution process in the problem space, and then obtains the optimal solution. The commonly used model of particle swarm optimization updating equation can be described as follows:

$$V_i^d(t+1) = \omega V_i^d(t) + c_1 r_1^d(pbest_i^d(t) - x_i^d(t)) + c_2 r_2^d(gbest^d - x_i^d(t)) \quad (1)$$

$$X_i^d(t+1) = X_i^d(t) + V_i^d(t+1) \quad (2)$$

where t is the current number of iterations, $d \in$ (1, 2, ..., D) and D is the dimension of the search space. $x_i(t)$ denotes the position of ith particle in tth iteration while $v_i(t)$ is the velocity vector of the ith particle in tth iteration. In order to control the particle's movement within the region of interest, the particle's velocity value in each dimension is limited to a value v_{max}^d. ω is the inertia weight introduced by shi and eberhart [2] (1998), which balances the exploration and exploitation capacity of the particles. A large inertia weight is beneficial to exploration, while a small inertia weight promotes exploitation [2]. r_1^d and r_2^d random values sampled from independent uniform distributions in the range $[0, 1]$. c_1 and c_2 are acceleration parameters. $pbest_i^d$ stores the best position achieved by the ith particle from the beginning to current and $gbest^d$ represents the best position found by the whole swarm so far.

The PSO algorithm is easy to implement and effective so that it has been widely applied in various optimization problems. However, it shows a flaw when solving some complicated multimodal problems. The algorithm usually traps into a solution, which may not be the optimal. Meanwhile, without appropriate mechanism, it cannot jump out of the deceptive position. It results to a lack of population diversity. But t he balance between exploration and exploitation abilities of this algorithm is important to find a high-quality solution.

Lots of PSO variants was proposed, and the existing PSO variants generally fall into two categories. The first category of PSO variants achieves improvement by changing the typical formula of (1) and (2). For example, Liang $et\ al.$ proposed a comprehensive learning particle swarm optimizer (CLPSO) [3] (2006), which uses all other particles' historical best information to update a particle's velocity. Qiang Lu and Qing Long Han proposed a model of finite-time particle swarm optimization algorithm [4] (2012), [15] (2014), which includes both continuous-time and discrete-time versions. By introducing a parameter and a nonlinear damping item into the algorithm, the exploitation of the discrete model of finite-time particle swarm optimization (DFPSO) algorithm is improved and it also make the DFPSO algorithm converge within a finite time interval. As pointed out by Lu and Han, this algorithm provides a flexible tool for solving optimization problems [4]. The other category aims to improve PSO by introducing heuristic or non-heuristic mechanisms, and there are various mechanisms including restarting mechanism, which has been widely used. For example, Keiji Tatsumi $et\ al.$ [5] (2009) proposed a restarting multi-swarm PSO (RMSPSO) algorithm, which uses two kinds of particles and multiple swarms including either kind of particles to search for solutions. In addition, this algorithm also uses a restarting strategy by random resetting the particle's velocity and position once the swarm falls into a trap. José García-Nieto and Enrique Alba proposed a restarting PSO with velocity modulation (RPSO-vm) algorithm [6] (2011). It uses a velocity modulation method to attract the particles within the region of interest. When the swarm falls into trap, it forces the particles to go to the best position or, if necessary, enables a random initialization . Tim Hendtlass [7] (2012) proposed a technique, which, when the swarm begins converging, disperses the particle to a position whose fitness is better than the average one, and when the swarm initializes in a good region, it will provide a better solution. Ling Lin $et\ al.$ [8] (2012) proposed a crown jewel defense strategy based on particle swarm optimization (PSOCJD), Which is used to relocate the global best position and initialize each particle's

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

personal best position once the particle's $pbest$ fitness has not improved in a certain period of time. For initializing particles in a good region, the algorithm provides a better solution in the next convergence. Although these algorithms improve the performance of PSO to a certain extent, It still remains a challenge to achieve fast convergence while maintaining population diversity during the search process.

In this paper, we propose a variant of PSO algorithm, called fast restarting particle swarm optimization (FRPSO) algorithm. The essence of this algorithm is that it provides a mechanism for fast convergence with a great exploitation capabilities and preserves proper population diversity at the same time. On the one hand, it can quickly converge to a solution no matter whether the solution is optimal or not. On the other hand, once the swarm gets into a trap, a restarting mechanism kicks in to make the swarm jump out the trap, and constantly looks for new and better solutions. Due to frequently searching and restarting, the swarm is able to search more promising regions and eventually finds the global optimum.

The rest of this paper is organized as follows: In Section II, the DFPSO algorithm will be briefly described. In Section III, details of the restarting strategy and the FRPSO algorithm will be elaborately described. The experiment on benchmark functions will be illustrated to demonstrate the performance of the proposed algorithm in Section IV. Finally, we will end this paper with a short conclusion in Section V.

Notation:Assumes that $sig(r)^a = sign(r)|r|^a$, where $0 < a < 1$, $r \in \mathbb{R}$ and $sign(\cdot)$ is the sign function.

II. DFPSO ALGORITHM

In this section, we will briefly describe the model of DFPSO algorithm which was introduced by Qiang Lu and Qing-Long Han in [4], [15]. Since the particles are independent in each dimension, without loss of generality, we set dimension $d = 1$ in the following.

The DFPSO algorithm derives from designing the continuous-time model of finite-time PSO (CFPSO)[4], [15]. Since the CFPSO algorithm has a well exploration capability, and uses the same discretization method as the generalized particle swarm optimization (GPSO) [9] to discretize the CFPSO, so that the exploiting capability of DFPSO was been further improved. Therefore, the DFPSO algorithm provides a well performance in purchasing the balance between exploration and exploitation abilities.

To be convenient, we assume $\alpha_1 = c_1 r_1$ and $\alpha_2 = c_2 r_2$ in Eqn. (1), and let $\alpha = \alpha_1 + \alpha_2$. Thus, we can regard α as a random number in the range [0,A], where A represents the upper bound of α. We also assume the oscillation center $p_i(t)$ as

$$p_i(t) = \frac{\alpha_1 pbest_i(t) + \alpha_2 gbest}{\alpha_1 + \alpha_2} \tag{3}$$

Then we can rewrite Eqn. (1) as

$$V_i(t+1) = \omega V_i(t) + \alpha(p_i(t) - x_i(t)) \tag{4}$$

The model of DFPSO can be given as

$$\begin{aligned} v_i(t+\Delta t) =& (1-\gamma(1-\omega)\Delta t) \\ & - \beta sig\Big(\gamma_1 x_i(t) - \gamma_2 p_i(t) \\ & + \gamma_3 v_i(t) + \gamma_4 p_i(t-\Delta t)\Big)^a \end{aligned} \tag{5}$$

$$x_i(t+\Delta t) = x_i(t) + v_i(t+\Delta t)\Delta t \tag{6}$$

with

$$\begin{aligned} \gamma_1 &= (1-\omega)\Delta t^{\frac{1-a}{a}} + \alpha \Delta t^{\frac{1}{a}} \\ \gamma_2 &= (1-\omega)\Delta t^{\frac{1-a}{a}} \\ \gamma_3 &= (1-\omega)\Delta t^{\frac{1}{a}} \\ \gamma_4 &= \alpha \Delta t^{\frac{1}{a}} \end{aligned}$$

where a, β, γ are parameters, and $0 < a < 1, \beta > 0$,and $0 < \gamma \leq 1$, $\beta sig(\cdot)^a$ is a nonlinear damping item [4], [10], [15]. It is obvious that if $\beta = 0$ and $\gamma = 1$, then the Eqn. (5) turns to Eqn. (1) and Eqn. (6) equals Eqn. (2). Therefore, the PSO algorithm can be regarded as a special case of the DFPSO algorithm. As pointed out by Qiang Lu and Qing Long Han in [4], The convergence speed of the swarm and the average oscillation magnitude of the position are controllable through adjusting the parameter β and γ . When β is increased, convergence time decreases, and when γ is reduced, the average oscillation magnitude increases.

III. METHODOLOGY

In this section, firstly, we will introduce a novel restarting strategy, which is inspired by the work on the designed Crown Jewel Defense (CJD) strategy [9]. Then we will propose the FRPSO algorithm which uses the restarting strategy based on DFPSO algorithm.

A. Restarting Strategy

We use $I_p(i)$ to record the increment of the ith particle's $pbest$ fitness value and use I_g to record the increment of the swarm's $gbest$ fitness value in a observation period T. T is a predefined number of iteration. We also use It to record the number of iterations in the process of operation. The calculation of $I_p(i)$ can be describe as

$$I_p(i) = \frac{fit(pbest_{t-1}) - fit(pbest_t)}{fit(pbest_{t-1})} \tag{7}$$

where $fit(pbest_t)$ stands for the personal best fitness value of tth iteration and $fit(pbest_{t-1})$ stands for the personal best fitness value of the previous iteration. The calculation of I_g can be describe as

$$I_g = \frac{fit(gbest_{t-1}) - fit(gbest_t)}{fit(gbest_{t-1})} \tag{8}$$

where $fit(gbest_t)$ means the global best fitness value of tth iteration and $fit(gbest_{t-1})$ denotes the global best fitness value of the previous iteration. It should be pointed out that $I_p(i)$ and I_g are calculated only if the the personal best of the particles and the global best of the swarm are updated, respectively.

We also use $RC_p(i)$ to record whether the personal best fitness value of the ith particle gets the improvement that is higher than the predefined threshold θ. RC_g is used to record whether the global best of the swarm is improved greater than the the threshold θ. The threshold θ is introduced to distinguish whether the fitness value improvement of the personal best or global best is negligible or not. When the swarm falls into trap, the particles oscillate around the local point and the improvement is so small that might be ignored. The calculation of $RC_p(i)$ can be described as

$$RC_p(i) = \begin{cases} RC_p(i) + 1 & if\ I_p(i) \geq \theta \\ RC_p(i) & if\ I_p(i) < \theta \end{cases} \quad (9)$$

and the calculation of RC_g is

$$RC_g = \begin{cases} RC_g + 1 & if\ I_g \geq \theta \\ RC_g & if\ I_g < \theta \end{cases} \quad (10)$$

Once a particle or the swarm gets trapped, the restating strategy is activated. The restarting strategy can be divided into two cases:

1) If the value of $RC_p(i)$ was not updated in an observation period T, then we regard that this particle has got into the trap, as shown in Fig.1 (a). In order to escape from the local optimal region, then initialize the personal best position, as shown in Fig.1 (b). Because the particle's flying is influenced by $pbest$, when initializing the value of $pbest$, the particle gets a new velocity, and thus it can jump out of the trap. Hence, the population diversity maintains. If the value of $RC_p(i)$ increases in period T, we will reset $RC_p(i)$. The improvement of $RC_p(i)$ represents the particle is searching. The pseudo code of this process is given in Algorithm 1.

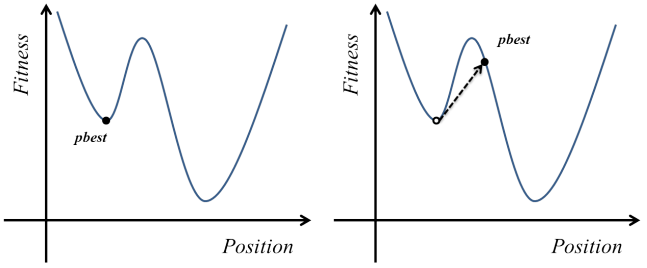


Fig. 1. (a) Particle falls into trap. (b) Random initialize the particle's $pbest$ to maintain population diversity.

Algorithm 1 Random Pbest()

```
for each particle in the swarm do
    if (It mod T == 0) and (RC_p(i) == 0) then
        reinitialize the pbest's postion of ith particle;
    end if
    if (It mod T == 0) and (RC_p(i)) ≠ 0 then
        RC_p(i) == 0;
    end if
end for
```

2) If the value of RC_g does not update in a observation period T, we would like to regard the swarm falling into a trap, as shown in Fig.2 (a). Therefore, it should take a measure to make the particles jump out of this region. This can be regarded as a special case of case one. Hence, we randomly reinitialize the velocity of the global best particle found and update its position according (6). Since the particle's moving is guided by $pbest$ and $gbest$, and the value of the $gbest$ affects all the particles in the swarm. If only $gbest$ is resetted, the swarm will still gets trapped. Therefore, we randomly reinitialize the position of the personal best of others particle, as shown in Fig.2 (b). Accordingly, if the value of RC_g increased in the observation period T, then we reset RC_g. This mechanism in case two is similar to CJD strategy [8]. The pseudo code of this process is given in Algorithm 2.

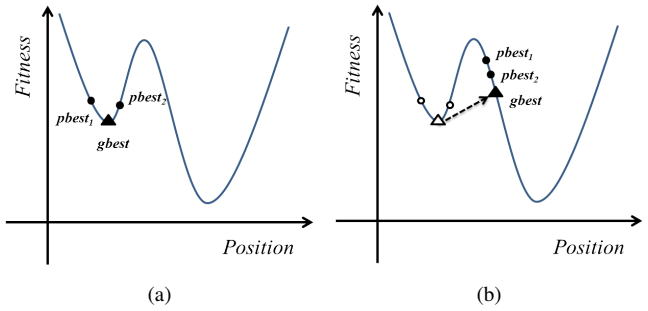


Fig. 2. (a)The swarm falls into trap. (b)After randomly initialize $gbest$ and $pbest$.The swarm reaches a new position.

Algorithm 2 Random Gbest()

```
if (It mod T == 0) and (RC_g == 0) then
    Reinitialize the gbest particle's velocity;
    Update gbest particle's position;
    for The rest particles of the swarm do
        Reinitialize the position of pbest;
    end for
end if
if (It mod T == 0) and (RC_g ≠ 0) then
    RC_g == 0;
end if
```

It should be pointed out that this restarting strategy is different from others strategies which are described in section I. This strategy constantly initializes particles to maintain population diversity once the improvements of a particle's $pbest$ are negligible. After that, the $gbest$ gets relocated and $pbest$ of all particles are discarded when a particle's $gbest$ is not improved in a certain period. And it is also designed to maintain the information that ever found and escape from where it has fallen.

B. FRPSO Algorithm

The restarting mechanism proposed above is based on the DFPSO algorithm, and the resultant algorithm is called FRPSO for short. Taking advantage of frequently speeding up the swarm to converge along with a greater exploitation capability and then jumping out of the trap, this algorithm preserves population diversity and provides a superior solution. The pseudo code of FRPSO is given in Algorithm 3.

Algorithm 3 FRPSO

BEGIN
Initialize the particle swarm;
Update $pbest$ $gbest$;
Initialize T,It,$RC_p(i)$,RC_g;
while $FEs < MaxFEs$ **do**
  for each particle in the swarm **do**
    Update particle's velocity according to (5);
    Update particle's position according to (6);
    Calculate particle fitness;
  end for
  if pbest fitness is improved **then**
    Update $pbest$ position and fitness value;
    Calculate $I_p(i)$ according to (7);
    Update $RC_p(i)$ according to (9);
  end if
  if gbest fitness is improved **then**
    Update $gbest$ position and fitness value;
    Calculate I_g according to (8);
    Update RC_g according to (10);
  end if
  $It = It + 1$;
  Random Pbest();
  Random Gbest();
end while
END

In order to demonstrate the benefits of mechanism used in FRPSO algorithm, we use the position diversity to measure the fast convergence and the great exploitation capacity of the FRPSO algorithm [11]-[13].

Position diversity measures the current particles' position distribution. Therefore, it can reflects particles' dynamics and gives the population diversity information of the swarm [13]. The calculation of position diversity can be given as

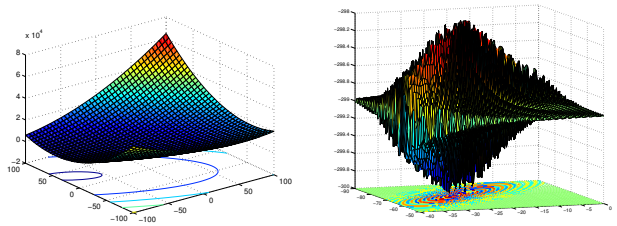
$$\overline{X}^d = \frac{1}{N}\sum_{i=1}^{N} x_i^d \tag{11}$$

$$P_{dis}^d = \frac{1}{N}\sum_{i=1}^{N}(x_i^d - \overline{X}^d)^2 \tag{12}$$

$$P_{div} = \frac{1}{D}\sum_{j=1}^{D} P_{dis}^d \tag{13}$$

where $\overline{X}_d$ means the average value of particles' current position on dth dimension. P_{dis}^d measures the Euclidean distance between all the particles' position and the average value of particles' current position for dth dimension [13]. P_{div} represents the whole swarms' current position diversity.

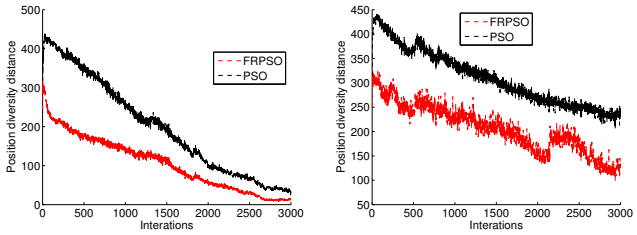
The experiment is performed on the single-model benchmark function f_2 and multimodal benchmark function f_{14} from [14], which can be illustrated in Fig.3 (a) and Fig.3 (b).



(a) (b)

Fig. 3. (a) Single-model benchmark function f_2. (b) Multimodal benchmark function f_{14}.

The population diversity curve on f_2 and f_{14} are illustrated in Fig.4 (a) and Fig.4 (b), respectively.



(a) (b)

Fig. 4. Population diversity of FRPSO algorithm and PSO algorithm on single-model and multimodal function (N=30, D=30). (a) Single model function f_2. (b) Multi-model function f_{14}.

From Fig.4, the following facts can be observed:

- From Fig.4 (a) and Fig.4 (b), the position diversity distance of the FRPSO algorithm is less than PSO algorithm on both of these functions, that means the FRPSO algorithm can make the particles get together quickly. In other words, the mechanism used in FPSO algorithm can speed up the particle to converge.
- From Fig.4 (b), the oscillation amplitude of the position diversity is more severer on FRPSO algorithm compared with PSO algorithm, it is because there are lots of locals position on multimodal function, and the restarting strategy in FRPSO algorithm constantly resetting $pbest$ or $pbest$ when the particle or the whole swarm falls into trap, that can make the particles jump out of the trap. This phenomenon also illustrates that the FRPSO algorithm has a greater capacity of exploitation.
- For the FRPSO algorithm, the oscillation amplitude of the position diversity is more severer in Fig.4 (b) compared with Fig4. (a), that means on single-model function, the restarting strategy less activated, since there is few local positions on it, but on multimodal function, the restarting strategy activities more frequently.

IV. EXPERIMENT

In this section, we illustrate the performance of the proposed FRPSO algorithm compared with other three representatives of advanced PSO algorithm.

A. Benchmark Functions

Twenty-five functions were selected from CEC 2005 [14] to test the performance of these algorithms. These functions include single-model , multimodal and hybrid composition functions which are listed in TABLE I. These functions are all minimizing problems and suitable for validating the effect of FRPSO algorithm.

TABLE I
BENCHMARK FUNCTIONS

Funtion	Type description
$f_1 - f_5$	Single-model function
$f_6 - f_{14}$	Multimodal function
$f_{15} - f_{25}$	Hybrid composition function

B. Parameter Settings for FRPSO

During the simulations, to be fair, the maximum fitness evaluation FE_{max} in each algorithm is equal to $5e + 04$. The particle size is set to 30, and the dimension is 30. The inertia weight ω linearly changes from 0.9 to 0.4 during the optimization process by using Enq.(14):

$$\omega = 0.9 - \frac{0.5 \times FE_s}{FE_{max}} \tag{14}$$

where FE_s is the current number of fitness evaluation. And the threshold value θ can affect the outcome radically. If it is too large, the swarm may be restarted frequently even though it does not converge currently; if it is too small, the swarm may be hardly restarted since the particles of the swarm oscillate around the local point, then it is difficult to reach the threshold. Therefore, the threshold value θ is set to $1e-6$. The observation period T is set to 4 so that it has enough time to determine whether particles fall into trap. The parameters of Eqn. (5) and Eqn. (6) are given in TABLE II,

TABLE II
PARAMETERS SETTING FOR Eqn.(5)and Eqn.(6)

Parameter	Value
alphabound	2.1
β	0.5
γ	0.8
Δt	0.5
a	0.5

It is notable that as discussed in section II, the PSO algorithm can be regarded as a special case of the DFPSO algorithm. And the parameters β will influence the convergent speed of the swarm, and in traditional PSO algorithm the parameter $\beta = 0$. Hence, in FRPSO algorithm we set $\beta = 0.5$, thus the convergence speed of FRPSO algorithm is faster than the traditional PSO algorithm. In addition, the parameters γ will influence the oscillation amplitude of particles during search, and in traditional PSO algorithm $\gamma = 1$. In FRPSO algorithm we set $\gamma = 0.8$, thus the average oscillation amplitude is increased.

In order to further illustrate the influence of β and γ on the convergence speed of position and the oscillation magnitude in FRPSO algorithm.

We use a generalized particle swarm optimization (GPSO) algorithm [9] as a comparison. For convenience, we set $p_i(t) = 0$. From Fig.5 and Fig.6, one can see that the convergence time of position decrease and the oscillation magnitude experiences a lightly increase in FRPSO algorithm comparing with GPSO algorithm.

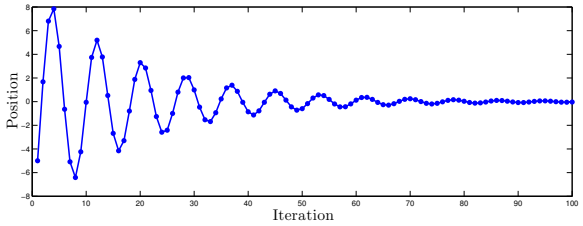


Fig. 5. The convergence trend of position of GPSO algorithm ($\omega = 0.8$, $a = 0.5$, $\alpha = 2.1$, $\beta = 0$, $\gamma = 1$, $\Delta t = 0.5$, $x_i(0) = -5$ $v_i(0) = 9$).

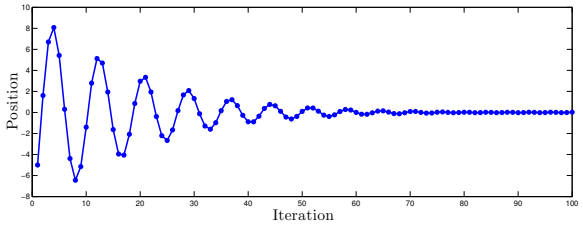


Fig. 6. The convergence trend of position of FRPSO algorithm ($\omega = 0.8$, $a = 0.5$, $\alpha = 2.1$, $\beta = 0.5$, $\gamma = 0.8$, $\Delta t = 0.5$, $x_i(0) = -5$ and $v_i(0) = 9$).

C. Comparisons with Other PSO

Three representative algorithms are used in the contrast experiment. These algorithms and their parameters are listed in Table III.

TABLE III
PARAMETERS SETTING FOR THE ALGORITHMS

Algorithm	Parameters	Reference
PSO	$N = 30, \omega = 0.9 \rightarrow 0.4, c1 = c2 = 2$	[2]
CLPSO	$N = 30, \omega = 0.9 \rightarrow 0.4, c1 = c2 = 2$	[3]
PSOCJD	$N = 30, \omega = 0.9 \rightarrow 0.4, c1 = c2 = 2$	[8]

where $\omega = 0.9 \rightarrow 0.4$ stands for ω linearly changes from 0.9 to 0.4 by use Eqn. (14). Each algorithm running 51 times independently.

The test results of all algorithms on the twenty-five functions are shown in Table IV. "mean" represents the mean value of performance among 51 runs, and "var" represents the standard deviation of these values. The bold data represents the best obtained value for the function. From Table IV, we can found:

- The performance of the proposed FRPSO algorithm is better than the other three representatives of advanced PSO algorithms on most of these functions, especially on $f_2 - f_6$, $f_{10} - f_{11}$, $f_{16} - f_{17}$, f_{21}, f_{23}, these function are covered by single-model, multimodal and hybrid composition functions.

TABLE IV
EXPERIMENTAL RESULTS OF ALL ALGORITHMS

Function		PSO	CLPSO	PSOCJD	FRPSO
f1	mean	**0.00e+000**	1.03e-003	**0.00e+000**	9.63e-006
	var	**0.00e+000**	5.44e-005	**0.00e+000**	4.73e-009
f2	mean	6.56e+001	3.99e+002	1.99e+002	**1.11e-004**
	var	2.19e+005	8.12e+006	2.03e+006	**6.31e-007**
f3	mean	5.10e+005	1.10e+006	8.80e+005	**3.93e+003**
	var	1.33e+013	6.21e+013	3.95e+013	**7.88e+008**
f4	mean	8.69e+002	7.11e+002	5.26e+002	**4.73e+002**
	var	3.85e+007	2.58e+007	1.41e+007	**1.14e+007**
f5	mean	9.88e+001	1.40e+002	1.54e+002	**9.08e+001**
	var	4.98e+005	9.99e+005	1.21e+006	**4.20e+005**
f6	mean	4.61e+000	1.15e+002	2.83e+000	**5.78e-001**
	var	1.09e+003	6.78e+005	4.08e+002	**1.70e+001**
f7	mean	**9.21e+001**	**9.21e+001**	**9.21e+001**	**9.21e+001**
	var	4.32e+005	4.32e+005	4.32e+005	**4.32e+005**
f8	mean	4.12e-001	**4.09e-001**	4.13e-001	4.13e-001
	var	8.66e+000	**8.55e+000**	8.69e+000	8.69e+000
f9	mean	2.32e+000	2.72e+000	**1.06e+000**	1.86e+000
	var	2.75e+002	3.77e+002	**5.74e+001**	1.76e+002
f10	mean	4.69e+000	3.19e+000	3.00e+000	**2.68e+000**
	var	1.12e+003	5.20e+002	4.59e+002	**3.65e+002**
f11	mean	5.74e-001	5.85e-001	6.67e-001	**2.85e-001**
	var	1.68e+001	1.75e+001	2.27e+001	**4.15e+000**
f12	mean	**1.75e+003**	6.44e+003	1.40e+004	2.59e+004
	var	**1.56e+008**	2.12e+009	9.98e+009	3.43e+010
f13	mean	1.91e-001	3.04e-001	**1.53e-001**	1.80e-001
	var	1.86e+000	4.71e+000	**1.20e+000**	1.65e+000
f14	mean	2.68e-001	2.60e-001	2.70e-001	**2.45e-001**
	var	3.66e+000	3.45e+000	3.73e+000	**3.05e+000**
f15	mean	6.64e+000	9.62e+000	**5.34e+000**	7.88e+000
	var	2.25e+003	4.72e+003	**1.45e+003**	3.17e+003
f16	mean	7.85e+000	4.92e+000	6.69e+000	**1.88e+000**
	var	3.14e+003	1.23e+003	2.28e+003	**1.80e+002**
f17	mean	6.06e+000	5.73e+000	6.06e+000	**4.99e+000**
	var	1.87e+003	1.68e+003	1.87e+003	**1.27e+003**
f18	mean	1.79e+001	**1.78e+001**	1.79e+001	**1.78e+001**
	var	1.63e+004	**1.62e+004**	1.63e+004	**1.62e+004**
f19	mean	1.80e+001	**1.78e+001**	**1.78e+001**	1.83e+001
	var	1.66e+004	1.62e+004	**1.61e+004**	1.70e+004
f20	mean	**1.77e+001**	1.79e+001	1.78e+001	1.95e+001
	var	**1.61e+004**	1.63e+004	1.62e+004	1.95e+004
f21	mean	2.16e+001	1.85e+001	9.97e+000	**9.80e+000**
	var	2.38e+004	1.74e+004	**5.07e+003**	4.90e+003
f22	mean	**1.83e+001**	1.85e+001	1.85e+001	2.20e+001
	var	**1.72e+004**	1.75e+004	1.75e+004	2.47e+004
f23	mean	2.17e+001	1.95e+001	**1.05e+001**	2.47e+001
	var	2.41e+004	1.95e+004	**5.59e+003**	3.11e+004
f24	mean	1.88e+001	1.90e+001	**3.92e+000**	**3.92e+000**
	var	1.80e+004	1.85e+004	**7.84e+002**	7.85e+002
f25	mean	2.16e+001	**1.98e+001**	2.02e+001	2.17e+001
	var	2.38e+004	**2.00e+004**	2.09e+004	2.39e+004

- The variances of FRPSO are also smaller than the three algorithms on most of these functions.

In order to further illustrate the advantages of the proposed algorithm, the average convergence traces of these algorithm are plotted in Fig. 7. We choose twelve typical functions which consists of single-model functions, multimodal functions and hybrid composition functions. The single mode functions include f_1, f_2, f_4 and f_5. The multimodal functions include f_{10}, f_{11}, f_{13} and f_{14}. The hybrid composition functions include f_{16}, f_{17}, f_{21} and f_{24}. The result indicates that:

- On the single-mode functions, the convergence speed of FPPSO algorithm is faster than the other algorithms. This benefits from the parameter β which decreases the convergence time.
- On multimodal functions, FRPSO maintains fast convergence. Once the swarm falls into trap, the restarting mechanism is activated, and the algorithm found a better solution. Especially on the function f_{11} and f_{14}.
- On hybrid composition functions, the convergence speed of FPPSO algorithm is still very faster.

V. CONCLUSIONS

In this paper, a FRPSO algorithm is proposed, which uses a novel restarting strategy based on DFPSO algorithm. This algorithm can speed up the swarm to converge. Once a particle gets trapped, it randomly initializes $pbest$. Similarly, once the swarm falls into a local solution, it random initializes $gbest$ and the others particles' $pbest$. Through quickly convergence and frequently restarting, the swarm can search more candidate solutions and eventually find the optimal one. Moreover, we use twenty-five benchmark functions to analyze the performance of the FRPSO algorithm. The experimental results also show that the proposed FRPSO algorithm performs better than the three other representatives of the advanced PSO algorithms on most of the twenty-five benchmark functions which consist of single-model, multimodal and hybrid composition problems. Nevertheless, it still has some work to do in the future. Such as, the performance of FRPSO algorithm on CEC 2013 should be tested, and the performance of the FRPSO on higher dimensionality are yet to be explored. Furthermore, The impacts of the parameters β and γ in the algorithm worth further discussing, and the parameters γ also can add into the restarting strategy. If the swarm renews frequently, then the value of parameter γ will be reduced. It improves the exploitation capacity. If the swarm updates slowly, then it increases the value of parameter γ. It improves the exploration capacity.

VI. ACKNOWLEDGEMENT

The authors would like to think Dr. Q. Lu at the department of Automation, Hangzhou Dianzi University, Hangzhou, China, for providing the source code of the DFPSO algorithm and some valuable suggestions on the research directions, which greatly facilitated our experiments. This work is supported by the National Natural Science Foundation of China (NSFC), under Grants No. 61272271, 61332008, 61174158, 61103072, 61103071 and 61103068. This program is also supported by the National Basic Research Program of China(973 Program) under Grant No. 2014CB340404, Natural Science Foundation Program of Shanghai under Grant No. 12ZR1434000, Fundamental Research Funds for the Central Universities under Grants No. 0800219201, Research Fund for the Doctoral Program of Higher Education of China (20110072120065) and the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry.

REFERENCES

[1] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *proc. of the Sixth Int. Symposium on Micromachine and Human Science,* Nagoya, Japan, pp. 39-43, 1995.

[2] R. C. Eberhart and Y. Shi, "Parameter selection in particle swarm optimization," in *Proceedings of the 7th International Conference on Evolutionary Programming*, San Diego, CA, USA, pp. 591-600, 1998.

[3] J. J. Liang, A. K. Qin, Ponnuthurai Nagaratnam Suganthan and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transaction on Evolutionary Computation*, pp. 281-295, 2006.

[4] Q. Lu and Q. L. Han, "A Finite-time Particle Swarm Optimization Algorithm," in *Proc. of the IEEE International Conference on Evaluation Computation,* Brisbane, Australia, pp. 1-8, 2012.

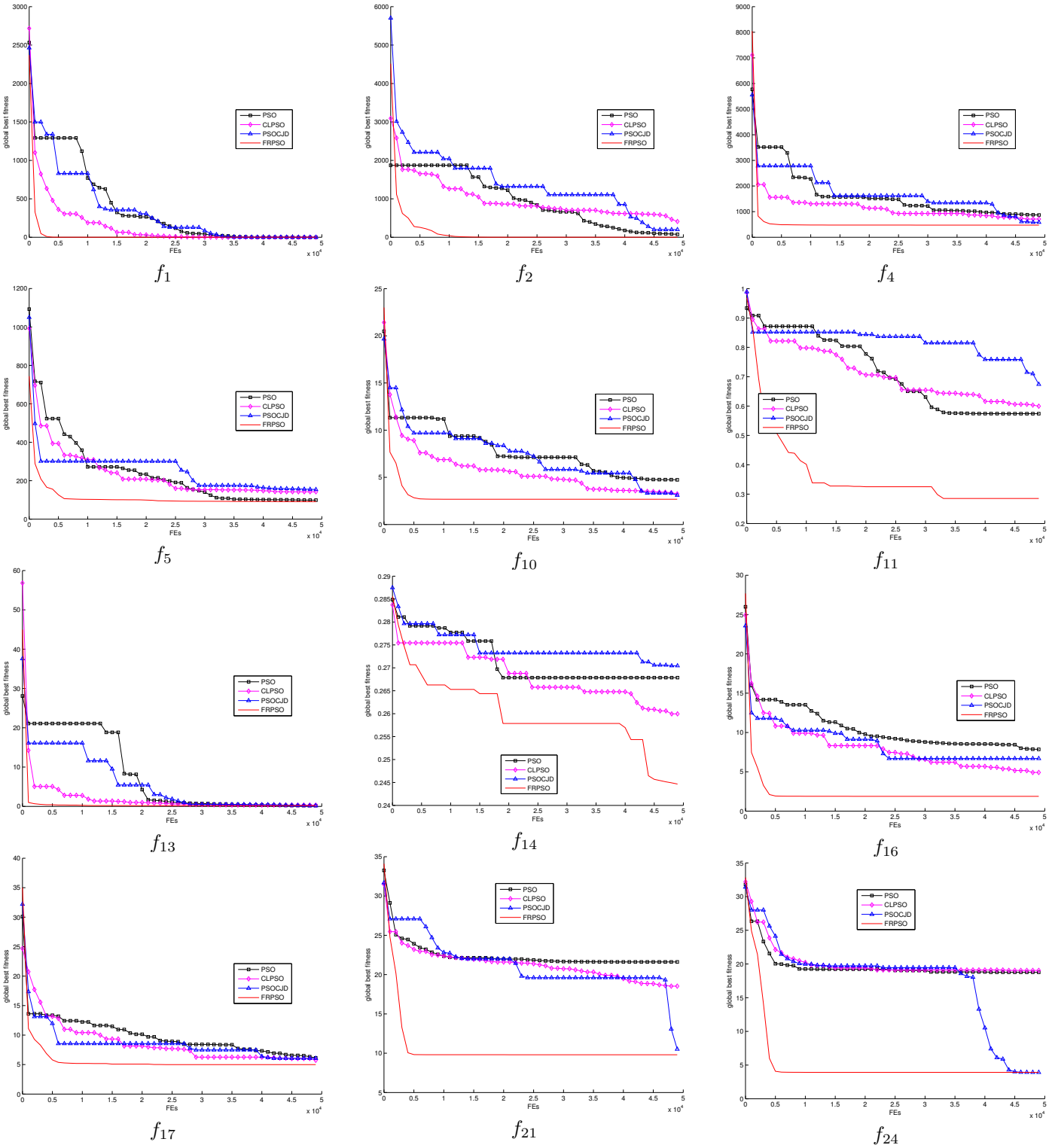


Fig. 7. Average convergence curve of PSO , CLPSO, PSOCJD and FRPSO over 51 runs on partial single-model , multimodal and hybrid composition functions with D =30.

[5] K. Tatsumi, T. Yukami and T. Tanino, "Restarting multi-type particle swarm optimization using an adaptive selection of particle type," in *IEEE International Conference on Systems, Man and Cybernetics*, pp. 923-928, October.2009.

[6] J. Gárca-Níeto and E. Alba, "Restart particle swarm optimization with velocity modulation: a scalability test," *Soft Comput*, vol. 15, pp. 2221-2232, 2011.

[7] T.Hendtlass, "Restarting Particle Swarm Optimization for Decptive Problems," in *Proc. of the IEEE International Conference on Evaluation Computation*, Brisbane, Australia, pp. 1-9, 2012.

[8] L. Lin, Z. Ji, S. He and Z. Zhu, "A Crown Jewel Defense Strategy Based Particle Swarm Optimization," in *Proc. of the IEEE International Conference on Evaluation Computation*, Brisbane, Australia, pp. 1-8, 2012.

[9] J. L. Fernández-Martínez, E. García-Gonzalo and J. P. Fernández-Alvarez, "Theoretical analysis of particle swarm trajectories through a mechnical analogy," *International Journal of Computational Intelligence Research*, vol. 4, no. 2, 2008.

[10] S. P. Bhat and D. S. Bernstein, "Finite-time stability of continuous autonomous systems," *SIAM Jounal of Control and Optimization*, vol. 38, no. 3, pp. 751-766, 2000.

[11] Y. Shi and R. Eberhart, "Population diversity of particle swarms," in *Proceedings of the 2008 Congress on Evolutionary Computation*, pp. 1063-1067, 2008.

[12] Y. Shi and R. Eberhart, "Monitoring of particle swarm optimization," *Frontiers of Computer Science*, vol. 3, no. 1, pp. 31-37, March 2009.

[13] S. Cheng, Y. Shi, and Q. Qin, "Population diversity based study on search information propagation in particle swarm optimization," in *Proceedings of 2012 IEEE Congress on Evolutionary Computation*, pp. 1272-1279, Brisbane, Australia, 2012.

[14] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger and S. Tiwari, "Problem definitions and evaluation criteria for the CEC 2005 special session on real-paramenter optimization,",2005. *SIAM Jounal of Control and Optimization*, vol. 38, no. 3, pp. 751-766, 2008.

[15] Q. Lu, Q. L. Han and S. R. Liu, "A finite-time particle swarm optimization algorithm for odor source localization," Inform. Sci.(2014), http://dx.doi.org/10.1016/j.ins.2014.02.010.