An Opposition-Based Learning Competitive Particle Swarm Optimizer

Jianhong Zhou, Wei Fang*, Xiaojun Wu, and Jun Sun
Department of Computer Science
School of IoT Engineering
Jiangnan University
Wuxi, China 214122
Email: fangwei@jiangnan.edu.cn

Shi Cheng
Division of Computer Science
University of Nottingham Ningbo, China
Email: shi.cheng@nottingham.edu.cn

***Abstract*—An opposition-based learning competitive particle swarm optimizer (OBL-CPSO) is proposed to address the problem of premature convergence in PSO. Two learning mechanisms have been employed in OBL-CPSO, which are competitive learning from competitive swarm optimizer (CSO) and opposition-based learning. In each iteration of OBL-CPSO, the competitive learning works among three randomly selected particles from the swarm and is followed by the comparison of fitness. The particle with the best fitness, denoted as the winner, is passed to the next iteration directly. The particle with the worst fitness learns from the winner and the particle with medium fitness takes the opposition-based learning for quickly exploiting the search space and then they are passed to the next iteration with updated positions and velocities. The performance of OBL-CPSO is evaluated on a set of benchmark functions and compared with PSO, six traditional PSO variants, and four state-of-art PSO variants. Experimental results show that OBL-CPSO has the best performance on the quality of solution and the convergence speed among the compared algorithms. The OBL-CPSO also has scalability to large scale global optimization problems from the experimental results on problems with 100, 300, and 500 dimensions. The OBL-CPSO is proposed to solve the learning problem of fuzzy cognitive maps (FCMs) which is a kind of complex networks and shows advantage over the compared algorithms for the problem.**

I. Introduction

Particle swarm optimizer (PSO) is a stochastic population-based optimization algorithm introduced by Kennedy and Eberhart [1], [2], which is inspired from the social cooperative and competitive behavior, such as bird flocking and fish schooling. In PSO, each particle is regarded as an individual in the population and flies around the multidimensional search space to explore the optimal solution. Each particle is represented by the position vector and velocity vector which are changed at every generation according to the cognitive and social experience. PSO shows powerful ability to find optimal solutions and is known for low computational complexity, ease of implementation and few parameters to be adjusted.

Since the proposal of PSO, it has attracted much research interest for the performance improvement. In general, convergence speed and global search ability are two considerable performance metrics of PSO algorithms. However, the performance of PSO usually suffers from the premature convergence, which is an inevitable problem in population-based algorithms and leads to the performance decreasing. Therefore, a large number of achievements are focused on the performance improvement of PSO through different techniques. In [3], [4], several interesting topological structures were proposed such as ring, circle, wheels, and von Neumann. Mendes et al. proposed a Fully Informed PSO (FIPS) where all the neighbors of a particle contribute to the velocity adjustment [5]. Liang et al. presented a Comprehensive Learning PSO (CLPSO) in [6] for multi-modal problems. Li designed a niching PSO based on a ring topology [7]. A distance-based locally informed PSO was proposed in [8] to solve multi-modal problems. An early study on the parameter can be found in [9] where the inertia weight was firstly introduced in PSO and later a linear decreasing approach was proposed in [10]. In [11], several time-varying methods were proposed for controlling two acceleration coefficients. The evolutionary operators such as mutation [12], crossover [13], and selection [14] are also incorporated with PSO. An interesting way to enhance the search ability of PSO is to sample the position of particles from a probability distribution including Levy distribution [15], Gaussian distribution [16], Exponential distribution [17]. A dynamic multi-swarm PSO is proposed in [18] where neighborhood structure dynamically changes for a higher swarm diversity. Many real-world problems solved by PSO and other EAs can be found in[19], [20], [21], [22].

In this paper, we also address the premature convergence and convergence speed of PSO and propose an Opposition-based Learning Competitive PSO (OBL-CPSO). In OBL-CPSO, two learning approaches are employed which are competitive learning and opposition learning. Here the competitive learning is an improved version of CSO [23] with more intelligent behavior on the swarm. The opposition-based learning is used to quickly exploit the search space,In addition, the algorithm was proposed and improved to be better used in practical problems. The rest of this paper is organized as follows. Section II introduces the framework of PSO. The principle of OBL-CPSO is described in Section III. Experimental results on the benchmark functions with 30, 100, 300, and 500 dimensions are presented and discussed in Section IV. The results of OBL-CPSO for solving fuzzy

978-1-5090-0623-6/16/$31.00

cognitive maps learning problem are also reported in Section IV. Conclusions are drawn in Section V.

II. PSO

In PSO, each particle in the swarm represents a potential solution of the problem with the position vector $X_i = [X_{i1}, X_{i2}, \cdots, X_{iD}]$ and velocity vector $V_i = [V_{i1}, V_{i2}, \cdots, V_{iD}]$, where D is the dimension of the problem. The personal historical best position of the *i*th particle is recorded and updated during the evolutionary procedure and is denoted as $P_i = [P_{i1}, P_{i2}, \cdots, P_{iD}]$. Among the swarm, the particle with the best fitness is regarded as the global best particle and is denoted as $G = [G_1, G_2, \cdots, G_D]$. In the initialization stage, the particles are generated with random positions and velocities in the limited searching space. And then with the evolutionary process, the particles adjust their velocities and positions continuously according to their own experience and swarm's experience as follows:

$$\begin{aligned} V_{id}(t+1) &= \omega \cdot V_{id} + c_1 \cdot r_{1d}(t) \cdot (P_{id}(t) - X_{id}(t)) \\ &+ c_2 \cdot r_{2d}(t) \cdot (G_d(t) - X_{id}(t)), \quad (1) \\ X_{id}(t+1) &= X_{id}c(t) + V_{id}(t+1), \quad (2) \end{aligned}$$

where ω is the inertia weight used to balance the global and local search abilities. The c_1 and c_2 are two positive constants, which are called self-learning factor and social learning factor respectively. Two independent random numbers r_{1d} and r_{2d} are uniformly distributed in the range of [0, 1]. From the velocity and position update equations, it can be seen that a particle's position is determined by its own previous velocity, the distance from the particle to its personal historical best position and the distance between the particle and the global best position in each iteration. The psedudocode of PSO is given in Algorithm 1.

Algorithm 1 Pseudocode of PSO

```
Generate the initial swarm with random positions and velocities in the problem space;
repeat
    for i = 1 to swarm size N do
        Update P_i and G;
        for j = 1 to D do
            Velocity update according to (1);
            Position update according to (2);
        end for
        Update the fitness value of particle i;
    end for
until termination criterion is met.
```

III. Opposition-based Learning Competitive PSO

A. Mechanism of Opposition-based Learning

Opposition-based learning (OBL) is a scheme from machine intelligence proposed in [24] and has been proven to be an effective technique to differential evolution (DE)[25]. In OBL, a candidate solution and its corresponding opposite one can be calculated in a single iteration. The opposite operator is an extension to current candidate solution and is expected to approach the optimal solution and then to achieve better fitness value in order to accelerate the convergence rate. Considering a real number x in the interval $[a, b]$, the opposite number $\tilde{x}$ of x is calculated as:

$$\tilde{x} = a + b - x. \quad (3)$$

It is very easy to extend the concept of opposite from one-dimensional space to multi-dimensional space. Let $P(x_1, x_2, \cdots, x_D)$ be a number in D-dimensional space with $x_1, x_2, \cdots, x_D$ are all real numbers and $x_i \in [a_i, b_i]$. Then the opposite point $\tilde{P}$ is defined by its coordinates $\tilde{x}_1, \tilde{x}_2, \cdots, \tilde{x}_D$ where

$$\tilde{x}_i = a_i + b_i - x_i. \quad (4)$$

The advantage of introducing Opposition-based learning is that, it can search in the current point and reverse point simultaneously, if fitness value in reverse point is better than the one in current point, the particle can directly jump to reverse points or its neighborhood and continue to search. This can broaden the solution space quickly, increase the probability of finding the global optimal solution.

B. The Competitive Learning Mechanism and OBL-CPSO

In PSO, the strong influence of global best particle is always regarded as the main reason for premature convergence. With the evolutionary process, a particle's personal historical best position is likely to have the same value as the global best particle and therefore the swarm's diversity decreases. In order to weaken or remove the influence of global best particle and personal historical best particle, a competitive swarm optimizer (CSO) is proposed in [23] where the swarm update by a consecutive random pairwise competition between particles within a single swarm. The competitive learning mechanism in CSO is very simple but efficient. After the pairwise competition, the particle with better fitness is the winner and another one is the loser. The winner particle is passed to the next iteration directly and the loser particle learn from the winner one. All the winners become the leaders of the swarm.

In this paper, we propose the opposition-based learning competitive PSO (OBL-CPSO) with a new competitive learning mechanism based on CSO and OBL, which is still simple but much more effective. In OBL-CPSO, each random competition is made among three particles within a single swarm. The three particles in each competition are randomly selected from the swarm and removed from the swarm after competition. In general, the result of competition will result in two particles with the best fitness and the worst fitness, termed as winner and loser respectively. The winner and loser evolve as those in CSO. That is, the winner is passed directly to the next iteration and the loser is passed to the next iteration after it learns from the winner. The particle with medium fitness, denoted as neutral, is designed to work under OBL and then passed to the next iteration, which helps to explore the search space. In each iteration, for a swarm size of N, $N/3$

TABLE I
BENCHMARK FUNCTIONS USED IN THIS PAPER

Mathematical Formulation	Search Space
$f_1(x) = \sum_{i=1}^{D} x_i^2$	[-100,100]
$f_2(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	[-10,10]
$f_3(x) = \sum_{i=1}^{D} (\sum_{j=1}^{i} x_j)^2$	[-100,100]
$f_4(x) = max_{i=1,...,D} \lvert x_i \rvert$	[-100,100]
$f_5(x) = \sum_{i=1}^{D-1} (100(x_i^2 - x_{i+1}^2) + (x_i - 1)^2)$	[-30,30]
$f_6(x) = \sum_{i=1}^{D} (\lfloor x_i + 0.5 \rfloor)^2$	[-100,100]
$f_7(x) = \sum_{i=1}^{D} i * x_i^4 + random[0,1)$	[-1.28,1.28]
$f_8(x) = \sum_{i=1}^{D} (x_i^2 - 10cos(2\pi x_i) + 10)$	[-5.12,5.12]
$f_9(x) = -20exp(-0.2\sqrt{\sum_{i=1}^{D} x_i^2/D}) - exp(\sum_{i=1}^{D} cos(2\pi x_i)/D) + 20 + e$	[-32,32]
$f_{10} = \sum_{i=1}^{D} x_i^2/4000 - \prod_{i=1}^{D} cos(x_i/\sqrt{i}) + 1$	[-600,600]
$f_{11} = \frac{\pi}{D}\{10(sin(\pi y_1))^2 + \sum_{i=1}^{D-1}(y_i - 1)^2[1 + 10(sin(\pi y_{i+1}))^2] + (y_D - 1)^2\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4)$ where $y_i = 1 + (x_i + 1)/4$, $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a \le x_i \le a \\ k(-x_i - a)^m & x_i < -a \end{cases}, i = 1, ..., D$	[-50,50]

competitions are required for all the particles to take part in the competition. After $N/3$ competitions, $2N/3$ particles update their velocities and positions which have higher efficiency than that in CSO. The loser [23] and neutral particles in OBL-CPSO update their velocities and positions by the following equations:

$$V_{ld}^k(t+1) = R_{1d}^k \cdot V_{ld}^k(t) + R_{2d}^k(t) \cdot (X_{wd}^k(t) - X_{ld}^k(t)) + \varphi \cdot R_{3d}^k(t) \cdot (\bar{X}_d^k(t) - X_{ld}^k(t)), \quad (5)$$

$$X_{ld}^k(t+1) = X_{ld}^k(t) + V_{ld}^k(t+1), \quad (6)$$

$$X_{nd}^k(t+1) = ub_d + lb_d - X_{nd}^k(t) + R_{4d}^k(t) \cdot X_{nd}^k(t), \quad (7)$$

where $X_{wd}^k(t), X_{ld}^k(t)$, and $X_{nd}^k(t)$ are the dth positions of the winner, loser and neutral in the kth round competition in iteration t, $V_{ld,k}$ is the dth velocity of the loser in the kth round competition in iteration t. The $R_{1d}^k(t), R_{2d}^k(t), R_{3d}^k(t)$, and $R_{4d}^k(t)$ are four random numbers within [0,1], φ is a manual parameter, $\bar{X}_d^k(t)$ is the mean position of all the particles, $ub_d and lb_d$ are the upper bound and lower bound in the dth search space. The psedudocode of OBL-CPSO can be found in Algorithm 2.

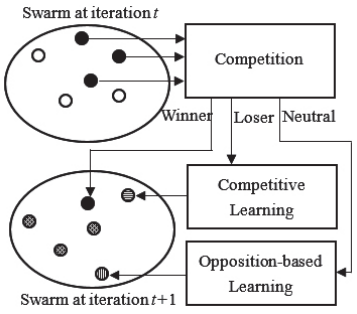

Fig. 1. Principles of OBL-CPSO

Algorithm 2 Pseudocode of OBL-CPSO

```
initialize P;
repeat
    shuffle(P); //shuffle particles form the swarm
    for k = 1:N/3 do
        r1 = P(k); //Select three particles from the swarm
        r2 = P(k + N/3);
        r3 = P(k + 2N/3);
        (w, n, l) = compete(r1,r2,r3); //Three particles compete according to results of evaluation
        Update the X_ld^k(t) according to (5) and (6);
        Update the X_nd^k(t) according to (7);
        Update the fitness value of the Loser and Neutral;
    end for
until termination criterion is met.
```

IV. EXPERIMENTAL STUDY

A. Experimental settings

In order to test the performance of the proposed OBL-CPSO, we make the comprehensive comparison among OBL-CPSO, PSO, PSO with constriction factor (PSO_Co)[26], FIPS [27], Gaussian Bare Bones PSO (GBBPSO)[28], Dynamic Multiple Swarm PSO (DMS_PSO) [18], CLPSO [6], quantum-behaved PSO (QPSO) [29], and four state-of-art PSO variants, which are random drift PSO (RDPSO) [30], Social Learning PSO (SL-PSO)[31], APSO-MAM[32], and CSO [23]. All the algorithms are used to solve a set of benchmark functions listed in Table I. The dimension D of all the 11 benchmark functions is 30 and the swarm size $N = M + D/10, M = 100$ as recommended in [33]. The maximum number of function evaluations(FEs) is set to be 200,000. According to the corresponding references, the parameter configurations for twelve algorithms are given in Table II.

TABLE II
PARAMETER SETTINGS FOR THE ALGORITHMS USED IN THE COMPARISON

Algorithms	Parameter settings
OBL-CPSO	$\varphi = 0.7 - 0.2$
RDPSO	$w : 0.9 - 0.3, c_1 = c_2 = 1.5$
QPSO	α=1.0-0.5
FIPS	$\chi = 0.729, \sum c_i = 4.1$
SL-PSO	$\alpha = 0.5, \beta = 0.01$
DMS_PSO	w:0.9-0.4, $c_1 = c_2 = 2.0, m = 3, R = 5$
PSO_Co	$\chi = 0.729, \sum c_i = 4.1$
PSO	$w : 0.9 - 0.4, c_1 = c_2 = 2.0$
GBBPSO	-
CLPSO	w:0.9-0.4, c=1.49445,m=7
CSO	$\varphi = 0$

B. Experimental results

The mean best fitness value and standard deviation out of 30 independent runs of each algorithm on each benchmark function are summarized in Table III. The comparison results show that OBL-CPSO gets outstanding performance among all the algorithms since it yields the best mean solutions on 8 benchmark functions. On solving functions $f_1, f_2, f_3, f_4, f_6, f_8$, and

TABLE III
COMPARISON RESULTS OF 12 ALGORITHMS ON 11 BENCHMARK FUNCTIONS (BEST RESULTS IN BOLD)

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}
OBL-CPSO	**7.27E-103**	**1.72E-55**	**7.01E-105**	**1.40E-53**	2.71E+01	**0.00E+00**	7.91E+00	**9.83E-102**	**4.44E-16**	**0.00E+00**	6.44E-02
	3.85E-102	**8.29E-55**	**2.54E-104**	**5.61E-53**	3.93E-01	**0.00E+00**	2.61E-01	**5.27E-101**	**4.30E-16**	**0.00E+00**	1.38E-02
RDPSO	1.52E-83	2.49E-27	4.43E-01	2.04E-04	3.98E+01	0.00E+00	**2.10E-03**	1.59E+01	6.45E-15	6.32E-03	**1.57E-32**
	8.35E-83	1.14E-26	4.57E-01	2.24E-04	4.19E+01	0.00E+00	**6.02E-04**	4.53E+00	2.07E-15	8.71E-03	**8.35E-48**
QPSO	7.49E-59	1.85E-34	4.16E+00	1.37E-04	3.72E+01	0.00E+00	2.20E-03	1.43E+01	2.06E+01	5.70E-03	2.72E-01
	1.90E-58	4.34E-34	3.79E+00	1.08E-04	2.68E+01	0.00E+00	7.58E-04	3.18E+00	5.51E-02	9.70E-03	1.13E-16
FIPS	1.30E-37	4.48E-20	3.09E+01	1.03E-01	**2.02E+01**	0.00E+00	1.09E-02	4.85E+01	2.10E+01	1.23E-08	1.57E-32
	1.08E-37	1.53E-20	1.17E+01	4.82E-02	**4.48E+00**	0.00E+00	2.50E-03	7.50E+00	4.61E-02	6.57E-08	8.35E-48
SL-PSO	3.27E-92	5.51E-48	6.03E-07	3.95E-25	2.86E+01	0.00E+00	1.38E-02	1.38E+01	5.51E-15	9.04E-04	1.57E-32
	3.49E-92	5.14E-48	5.44E-07	3.90E-25	2.19E+01	0.00E+00	3.00E-03	3.64E+00	1.45E-15	2.86E-03	5.57E-48
DMS_PSO	2.69E+02	5.04E+00	1.56E+03	6.96E+00	1.42E+04	1.84E+02	3.70E-03	6.62E+01	2.10E+01	6.62E-01	8.26E-06
	3.06E+02	3.18E+00	1.06E+03	4.90E+00	3.10E+04	2.04E+02	2.30E-03	4.72E+01	8.55E-02	3.63E-01	1.04E-05
PSO_Co	1.03E-48	1.54E-26	8.46E+00	8.78E-05	3.98E+01	1.00E-01	8.80E-03	4.35E+01	2.10E+01	1.58E-02	1.57E-32
	2.20E-48	4.01E-26	4.63E+01	1.13E-04	3.41E+01	3.05E-01	4.90E-03	1.35E+01	3.61E-01	1.55E-02	8.35E-48
PSO	1.89E-13	7.47E-10	1.76E+03	4.24E+00	8.93E+01	6.67E-02	5.18E-02	2.90E+01	2.11E+01	1.74E-02	6.79E-20
	3.57E-13	1.08E-09	2.47E+03	2.97E+00	8.90E+01	2.54E-01	2.86E-02	1.36E+01	1.00E-01	1.70E-02	1.59E-19
GBBPSO	3.68E-44	2.22E-31	9.63E+00	4.85E-01	5.70E+01	3.33E-02	8.70E-03	4.06E+01	2.09E+01	5.70E-03	1.78E-32
	6.89E-44	3.06E-31	8.61E+00	4.28E-01	6.97E+01	1.83E-01	3.00E-03	1.06E+01	1.84E-01	8.60E-03	5.28E-33
CLPSO	4.26E-05	6.69E-04	3.28E+03	1.06E+01	1.21E+02	0.00E+00	7.20E-03	3.85E+00	2.03E+01	1.28E-05	8.46E-11
	9.75E-06	1.15E-04	6.32E+02	8.61E-01	2.28E+01	0.00E+00	1.60E-03	1.32E+00	4.11E-02	1.08E-05	2.00E-11
CSO	1.88E-85	7.53E-43	5.27E-03	4.96E-03	2.55E+01	0.00E+00	3.00E-03	8.42E+00	2.66E-15	0.00E+00	1.57E-32
	2.06E-85	4.57E-43	6.43E-03	8.67E-03	1.02E+01	0.00E+00	8.15E-04	1.93E+00	2.63E-15	0.00E+00	5.57E-48

TABLE IV
THE *t*-TEST RESULTS OF COMPARISON AMONG OBL-CPSO AND THE OTHER 11 ALGORITHMS

t-test	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	*b/w/e/gm*
OBL-CPSO vs RDPSO	=	=	+	+	=	=	-	+	+	+	-	5/2/4/3
OBL-CPSO vs QPSO	+	+	+	+	+	=	-	+	+	+	-	8/2/1/6
OBL-CPSO vs FIPS	+	+	+	+	-	=	-	+	+	=	-	6/3/2/3
OBL-CPSO vs SL-PSO	+	+	+	+	=	=	-	+	+	=	-	6/2/3/4
OBL-CPSO vs DMS_PSO	+	+	+	+	+	+	-	+	+	+	-	9/2/0/7
OBL-CPSO vs PSO_Co	+	+	=	+	+	=	-	+	+	+	-	7/2/2/5
OBL-CPSO vs PSO	+	+	+	+	+	=	-	+	+	+	-	8/2/1/6
OBL-CPSO vs GBBPSO	+	+	+	+	+	=	-	+	+	+	-	8/2/1/6
OBL-CPSO vs CLPSO	+	+	+	+	+	=	-	+	+	+	-	8/2/1/6
OBL-CPSO vs CSO	+	+	+	+	=	=	-	+	+	=	-	6/2/3/4

f_{10}, the results by OBL-CPSO have very high precision. FIPS, GBBPSO, and RDPSO show better performance on f_5, f_7, and f_{11} respectively. Clearly, OBL-CPSO has obvious advantage on both uni-modal and multi-modal functions.

In order to determine whether OBL-CPSO gets statistically significant improvement, the *t*-test with a 95% significance level has also been carried out between every two algorithms. Table IV presents the *t*-test values on every function of the unpaired two-tailed test between the algorithms. In Table IV, the signatures '+', '-', and '=' represent that OBL-CPSO has significantly better performance than, significantly worse performance than, and equal performance to the compared algorithm. The term '*b/w/e*' shows the statistical results on all the benchmark functions that OBL-CPSO is significantly better than, significantly worse than, and equal to the compared algorithm, and the value of 'gm' shows the difference between '*b*' and '*w*' in order to reveal an overall comparison between two algorithms [34]. From the statistical results in Table IV, the performance of OBL-CPSO is better than all the other 11 algorithms,this confirms the advantage of OBL-CPSO.

Fig. 2 presents the comparison among the algorithms on the convergence characteristics for solving 11 benchmark functions. For all the functions, OBL-CPSO shows the fastest convergence speed on all the functions which can be attributed to the opposition-based learning mechanism.The source code of OBL-CPSO is given on the website http://cn.mathworks.com/matlabcentral/fileexchange/56418-obl-cpso.

C. Scalability to 100, 300, and 500 dimensions

For further evaluating the performance of OBL-CPSO on higher dimensional problems, we extend the benchmark functions to 100, 300, and 500 dimensions and solve them by OBL-CPSO. The parameters for OBL-CPSO are the same as those in the former subsection. Table V summarises the average optimization results of OBL-CPSO on 100, 300, and 500 functions after 30 independent runs. The performance of algorithm usually deteriorates as the problem's dimension increases which is known as 'curse of dimensionality'. However, from the results in Table V, it is obviously that OBL-CPSO still holds good performance on higher dimensional benchmark functions. For functions f_2, f_3, and f_4, they are all unimodal inseparable functions and hard to be solved with good precision under higher dimensionality. OBL-CPSO also

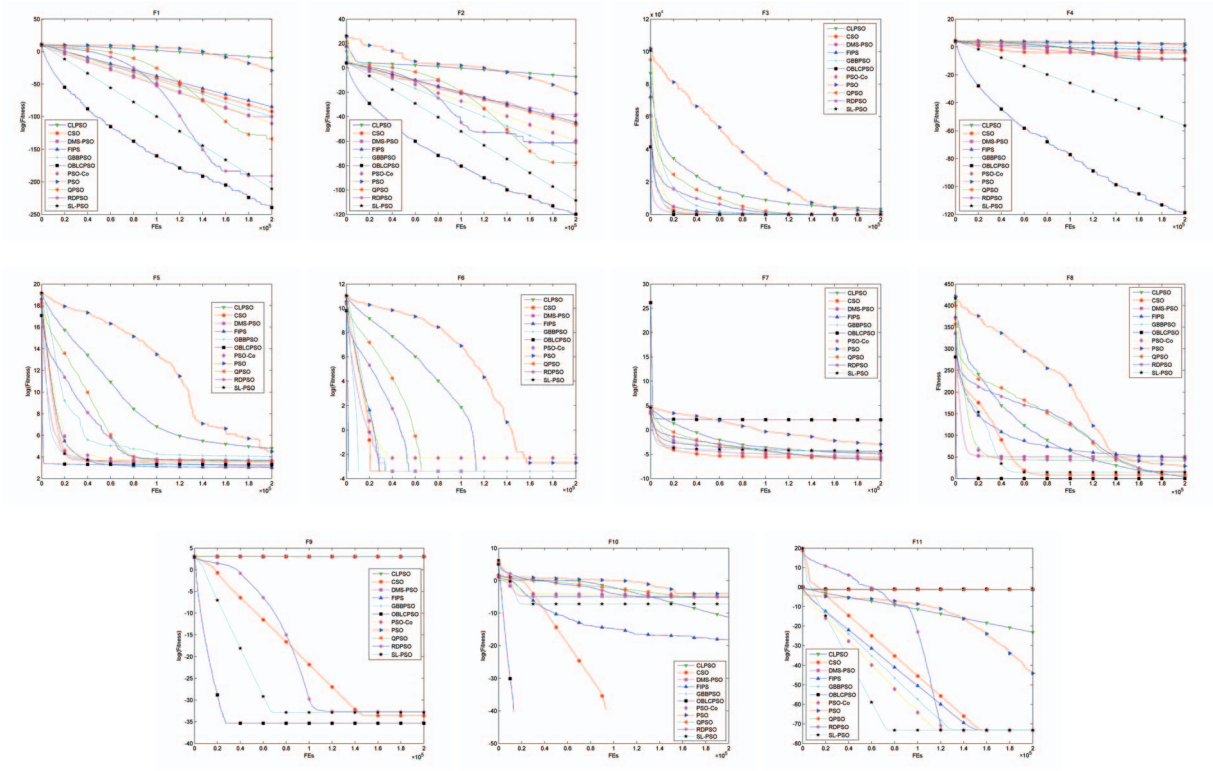

Fig. 2. Comparison of convergence curves of 12 algorithms on 11 benchmark functions

gets good solutions with high precision on functions f_6, f_8, f_9, and f_{10} with 100, 300, and 500 dimensions.

TABLE V
RESULTS OF OBL-CPSO ON 11 BENCHMARK FUNCTIONS WITH 100, 300, AND 500 DIMENSIONS

	30D	100D	300D	500D
f_1	7.27E-103	1.33E-101	2.47E-91	8.98E-96
f_2	1.72E-55	4.48E-52	1.08E-48	5.71E-50
f_3	7.01E-105	6.18E-98	3.18E-95	1.20E-93
f_4	1.41E-53	3.22E-50	5.78E-49	8.94E-50
f_5	2.71E+01	9.73E+01	2.97E+02	4.97E+02
f_6	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_7	7.91E+00	3.65E+01	1.26E+02	2.19E+02
f_8	9.83E-102	5.56E-98	5.39E-93	1.61E-96
f_9	4.44E-16	4.44E-16	4.44E-16	4.44E-16
f_{10}	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_{11}	6.44E-02	4.25E-01	6.48E-01	7.46E-01

D. Application in fuzzy cognitive maps learning problem

Fuzzy cognitive maps (FCMs) are useful tools for constructing models for complex systems and have several advantages than traditional modeling approaches. An FCM is a directed fuzzy graph with a set of nodes and weighted edges. Fig. 3 shows an FCM with five concept nodes and weighted edges. The nodes represent real-world concepts such as values, goals, etc. and the weighted and signed edges stands for the relationship between two connected concepts (nodes). The values of concepts and the weight matrix for the edges are important to the performance of a given FCM and can be determined by several kinds of learning algorithms instead of experts. Here we use the proposed OBL-CPSO to train an FCM from real-life data [35] for evaluating the algorithm's performance. There are nine concepts in the FCM. The initial values of concepts of the FCM for five response sequences are given in Table VI.

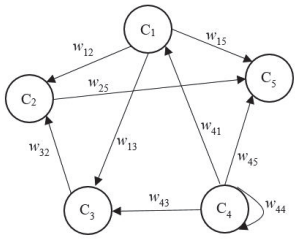

Fig. 3. An example of FCM with five concept nodes

TABLE VI
THE VALUES OF CONCEPTS FOR THE FCM WITH FIVE RESPONSE SEQUENCES

C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9
0.24	0.48	0.2	0.1	0.15	0.4	0.00	0.07	0.61
0.65	0.5	0.2	0.4	0.45	0.5	0.51	0.4	0.53
0.74	0.48	0.2	0.41	0.51	0.4	0.54	0.48	0.62
0.73	0.48	0.2	0.4	0.5	0.4	0.55	0.47	0.64
0.72	0.48	0.2	0.39	0.5	0.4	0.55	0.47	0.64

TABLE VII
EXPERIMENTAL RESULTS OF OBL-CPSO AND THE OTHER 11 ALGORITHMS ON A REAL LIFE FCM

Algorithms	$Data\_Error$		$Out\_of\_Sample\_Error$	
	Mean	Std.	Mean	Std.
CLPSO	3.24E-02	1.10E-03	1.22E-01	5.90E-03
DMS_PSO	2.81E-02	1.90E-03	1.19E-01	1.06E-02
FIPS	2.56E-02	1.70E-03	1.03E-01	2.30E-03
GBBPSO	7.41E-02	1.78E-02	2.27E-01	3.73E-02
PSO_Co	4.91E-02	1.11E-02	1.63E-01	4.94E-02
PSO	6.91E-02	1.22E-02	2.04E-01	7.13E-02
SL-PSO	2.84E-02	5.74E-04	1.03E-01	2.10E-03
QPSO	2.88E-02	3.50E-03	1.23E-01	1.63E-02
CSO	2.27E-02	5.82E-04	8.75E-02	1.60E-03
RDPSO	3.86E-02	8.40E-03	1.62E-01	2.21E-02
OBL-CPSO	**1.64E-02**	**5.67E-05**	**7.90E-02**	**9.56E-04**

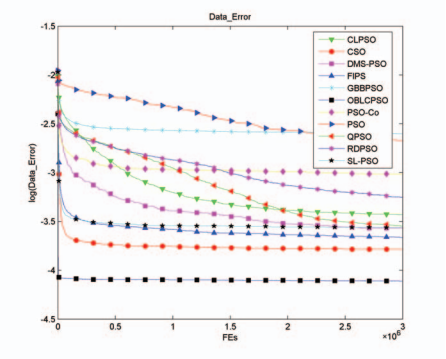

Fig. 4. Comparison of convergence curves on $Data\_Error$ of 12 algorithms for a real life FCM

The objective function for FCM learning problem is to evaluate the difference between the given response sequences and the generated response sequences, which can be defined as [35],

$$Data\_Error(W) = \frac{\sum_{\substack{1\le t\le N_t-1\\1\le n\le N_n\\1\le s\le N_s}} (C_n^t(s) - \hat{C}_n^t(s))^2}{N_n N_s (N_t - 1)}, \quad (8)$$

$$C_n^{t+1} = g(\sum_{j=1}^{N_n} w_{ji} C_j^t), i = 1, 2, \cdots, N_n, \quad (9)$$

$$g(x) = \frac{1}{1 + e^{-5x}} \quad (10)$$

where $N_n = 9$ is the number of concepts, $N_t = 10$ is the number of time points used in the response sequences, $N_s = 5$ is the number of response sequences, $C_n^t(s)$ and $\hat{C}_n^t$ are the tth state values of the nth node in the sth given and generated response sequences respectively. The values of concepts is in the range of [0, 1] and the weights of the edges are between [-1, 1]. The function $Out\_of\_Sample\_Error$ given in (11) is used to evaluate the ability of the algorithm when it handling the over-fitting problems [35]. For calculating the value of $Out\_of\_Sample\_Error$, ten other randomly generated initial state vectors are used.

$$Out\_of\_Sample\_Error(W) = \frac{1}{N_s N_n (N_t - 1)} \cdot \sum_{s=1}^{N_s} \sum_{t=1}^{N_t-1} \sum_{n=1}^{N_n} |C_n^t(s) - \hat{C}_n^t(s)|. \quad (11)$$

In order to apply OBL-CPSO to solve the FCM learning problem, the generated weight matrix is translated to a vector and represented as a particle in OBL-CPSO. The particle is real-coded and treated as a candidate solution. The dimension of a particle in this example is 81. The population size is 100 and the maximum number of FEs is 300,000. The other 11 algorithms used in subsection IV-B are also taken to solve the FCM learning problem for comparison. The parameters of these algorithms are the same as those shown in Table II. All the 12 algorithms run 30 independent times and the average values of $Data\_Error$ and $Out\_of\_Sample\_Error$ are recorded in Table VII. Fig. 4 and Fig. 5 draw the convergence curves of the algorithms for solving the learning problem of a real life FCM. From the results in Table VII, OBL-CPSO outperforms the other algorithms both on $Data\_Error$ and $Out\_of\_Sample\_Error$. In terms of the comparison on convergence speed, OBL-CPSO is the best one among all the compared algorithms and followed by CSO. The advantages of robustness of OBL-CPSO can also be confirmed from the values of standard deviation as given in Table VII.

V. CONCLUSION

In this paper, an opposition-based learning competitive particle swarm optimizer is proposed by introducing the competitive learning mechanism and opposition-based learning mechanism. The competitive learning mechanism is taken on three particles from the swarm, which are selected randomly. They distinguish with each other by comparing their fitness values. The particle with the worst fitness updates its position and velocity by learning from the particle with the best fitness. The particle with medium fitness updates itself by taking the opposition-based learning. The competitive learning cooperated with opposition-based learning in OBL-CPSO helps the algorithm to solve the problems with good exploration and exploitation abilities. Experimental results on benchmark functions show that OBL-CPSO outperforms the the other 11 compared PSO variants. The OBL-CPSO is proposed to solve

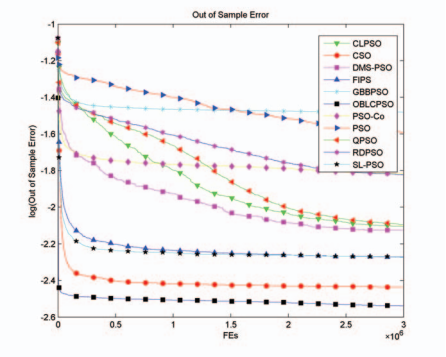

Fig. 5. Comparison of convergence curves on $Data\_Error$ of 12 algorithms for a real life FCM

the learning problem of FCMs and it shows good performance on this problem. We will further study the convergence proof of OBL-CPSO and its performance on large scale global optimization problems. Much More optimization problems in complex networks will also be considered to be solved by OBL-CPSO.

ACKNOWLEDGMENT

The author gratefully acknowledges the support of K.C.Wong Education Foundation, Hong Kong. This work was partially supported by the National Natural Science foundation of China (Grant Nos. 61105128, 61170119, 61373055), the Natural Science Foundation of Jiangsu Province, China (Grant No. BK20131106), the Postdoctoral Science Foundation of China (Grant No. 2014M560390), the Fundamental Research Funds for the Central Universities, China (Grant No. JUSRP51410B), Six Talent Peaks Project of Jiangsu Province (Grant No. DZXX-025), the PAPD of Jiangsu Higher Education Institutions, China.

REFERENCES

[1] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of IEEE International Conference on Neural Networks*, vol. 4, 1995, pp. 1942–1948.

[2] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *Proceedings of the IEEE International Conference on Evolutionary Computation*.

[3] P. N. Suganthan, "Particle swarm optimiser with neighbourhood operator," in *Proceedings of the 1999 Congress on Evolutionary Computation*, vol. 3, 1999, p. 1962.

[4] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," vol. 2, 2002, pp. 1671–1676.

[5] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: Simpler, maybe better," *Ieee Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 204–210, 2004.

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

[7] L. Xiaodong, "Niching without niching parameters: Particle swarm optimization using a ring topology," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 1, pp. 150–169, 2010.

[8] B. Y. Qu, P. N. Suganthan, and S. Das, "A distance-based locally informed particle swarm model for multimodal optimization," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 3, pp. 387–402, 2013.

[9] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," in *Proceedings of the 7th International Conference on Evolutionary Programming VII*, vol. 1447. New York: Springer-Verlag London, UK, 1998, pp. 591–600.

[10] ——, "Empirical study of particle swarm optimization," in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 3.

[11] M. Hu, W. T., and W. J. D., "An adaptive particle swarm optimization with multiple adaptive methods," *Evolutionary Computation, IEEE Transactions on*, vol. 17, no. 5, pp. 705–720, 2013.

[12] Y. V. Pehlivanoglu, "A new particle swarm optimization method enhanced with a periodic mutation strategy and neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 3, pp. 436–452, 2013.

[13] A. P. Engelbrecht, "Particle swarm optimization with crossover: a review and empirical analysis," *Artificial Intelligence Review*, pp. 1–35, 2015.

[14] A. Askarzadeh and L. d. S. Coelho, "Using two improved particle swarm optimization variants for optimization of daily electrical power consumption in multi-chiller systems," *Applied Thermal Engineering*, vol. 89, pp. 640–646, 2015.

[15] T. J. Richer and T. M. Blackwell, "The levy particle swarm," in *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, 2006, pp. 808–815.

[16] R. A. Krohling, "Gaussian swarm: a novel particle swarm optimization algorithm," in *2004 IEEE Conference on Cybernetics and Intelligent Systems*, vol. 1, 2004.

[17] F. B. Sun, Jun and W. Xu, "Particle swarm optimization with particles having quantum behavior," in *IEEE Congress on Evolutionary Computation*, 2004, pp. 325–331.

[18] J. J. Liang and P. N. Suganthan, "Dynamic multi-swarm particle swarm optimizer," in *Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE*, 2005, pp. 124–129.

[19] A. Khmelev and Y. Kochetov, "A hybrid local search for the split delivery vehicle routing problem," *International Journal of Artificial Intelligence*, vol. 13, no. 1, pp. 147–164, 2015.

[20] L. Feng, Y. S. Ong, A. H. Tan, and I. W. Tsang, "Memes as building blocks: a case study on evolutionary optimization + transfer learning for routing problems," *Memetic Computing*, vol. 7, no. 3, pp. 159–180, 2015.

[21] R.-E. Precup, R.-C. David, E. M. Petriu, S. Preitl, and M.-B. Radac, "Fuzzy logic-based adaptive gravitational search algorithm for optimal tuning of fuzzy-controlled servo systems," *Control Theory & Applications, Iet*, vol. 7, no. 1, pp. 99–107, 2013.

[22] J. Liu, H. A. Abbass, D. G. Green, and W. Zhong, "Motif difficulty (md): a predictive measure of problem difficulty for evolutionary algorithms using network motifs," *Evolutionary computation*, vol. 20, no. 3, pp. 321–347, 2012.

[23] R. Cheng and Y. Jin, "A competitive swarm optimizer for large scale optimization," *IEEE transactions on cybernetics*, vol. 45, no. 2, pp. 191–204, 2014.

[24] H. R. Tizhoosh, "Opposition-based learning: A new scheme for machine intelligence," in *International Conference on Computational Intelligence for Modelling, Control and Automation*, vol. 1, 2005, Conference Proceedings, pp. 695–701.

[25] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition-based differential evolution," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 64–79, 2008.

[26] M. Clerc and J. Kennedy, "The particle swarm - explosion, stability, and convergence in a multidimensional complex space," *Ieee Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[27] J. Kennedy and R. Mendes, "Neighborhood topologies in fully-informed and best-of-neighborhood particle swarms," in *Proceedings of the 2003 IEEE International Workshop on Soft Computing in Industrial Applications*, 2003, pp. 45–50.

[28] J. Kennedy, "Bare bones particle swarms," in *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, 2003, pp. 80–87.

[29] W. Fang, J. Sun, Y. R. Ding, X. J. Wu, and W. Xu, "A review of quantum-behaved particle swarm optimization," *IETE Technical Review*, vol. 27, no. 4, pp. 336–348, 2010.

[30] J. Sun, X. Wu, V. Palade, W. Fang, and Y. Shi, "Random drift particle swarm optimization algorithm: convergence analysis and parameter selection," *Machine Learning*, vol. 101, no. 1-3, pp. 345–376, 2015.

[31] R. Cheng and Y. Jin, "A social learning particle swarm optimization algorithm for scalable optimization," *Information Sciences*, vol. 291, no. 0, pp. 43–60, 2015.

[32] M. Hu, T. Wu, and J. D. Weir, "An adaptive particle swarm optimization with multiple adaptive methods," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 5, pp. 705–720, 2013.

[33] A. Auger and N. Hansen, "A restart cma evolution strategy with increasing population size," in *The 2005 IEEE Congress on Evolutionary Computation*, vol. 2, Conference Proceedings, pp. 1769–1776.

[34] Z.-H. Zhan, J. Zhang, Y. Li, and C. H. S. H., "Adaptive particle swarm optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 39, no. 6, pp. 1362–1381, 2009.

[35] Y. Chi and J. Liu, "Learning of fuzzy cognitive maps with varying densities using a multi-objective evolutionary algorithm," *IEEE Transactions on Fuzzy Systems*, vol. PP, no. 99, pp. 1–1, 2015.