Nat Comput (2010) 9:659–682
DOI 10.1007/s11047-009-9174-4

Discrete and continuous optimization based on multi-swarm coevolution

Hanning Chen · Yunlong Zhu · Kunyuan Hu

Published online: 24 December 2009

Abstract This paper presents a novel Multi-swarm Particle Swarm Optimizer called PS^2O, which is inspired by the coevolution of symbiotic species in natural ecosystems. The main idea of PS^2O is to extend the single population PSO to the interacting multi-swarms model by constructing hierarchical interaction topology and enhanced dynamical update equations. With the hierarchical interaction topology, a suitable diversity in the whole population can be maintained. At the same time, the enhanced dynamical update rule significantly speeds up the multi-swarm to converge to the global optimum. The PS^2O algorithm, which is conceptually simple and easy to implement, has considerable potential for solving complex optimization problems. With a set of 17 mathematical benchmark functions (including both continuous and discrete cases), PS^2O is proved to have significantly better performance than four other successful variants of PSO.

Keywords Multi-swarm · Coevolution · Symbiosis · Hierarchical interaction topology · PSO · PS^2O

1 Introduction

Swarm Intelligence (SI) is an innovative artificial intelligence technique for solving complex optimization problems. This discipline is inspired by the collective behaviors of social animals such as fish schools, bird flocks, and ant colonies. In SI systems, there are many simple individuals who can interact locally with one another and with their

H. Chen (✉) · Y. Zhu · K. Hu
Key Laboratory of Industrial Informatics, Shenyang Institute of Automation, Chinese Academy of Sciences, Faculty Office III, Nanta Street 114#, Dongling District, 110016 Shenyang, China
e-mail: chenhanning@sia.cn

Y. Zhu
e-mail: ylzhu@sia.cn

K. Hu
e-mail: hukunyuan@sia.cn

environments. Although such systems are decentralized, local interactions between individuals lead to the emergence of global behavior and properties.

In recent years, many algorithmic SI methods were designed to deal with practical problems (Dorigo and Gambardella 1997; Karaboga and Basturk 2008; Passino 2002). Among them, the most successful is Particle Swarm Optimization (PSO) that drew inspiration from the biological swarming behaviors observed in flocks of birds, schools of fish, and even human social behavior (Eberhart and Kennedy 1995; Kennedy and Eberhart 2001). PSO is a population-based optimization tool, which could be implemented and applied easily to solve various function optimization problems. As a problem-solving technique, the main strength of PSO is its fast convergence, which compares favorably with Evolutionary Algorithms (EAs) and other global optimization algorithms (Koza 1992; Yao et al. 1999; Bäck and Schwefel 1995; Holland 1975). However, when solving complex multimodal problems, PSO suffer from the following drawback (Angeline 1998): as a population evolves, all individuals suffer premature convergence to the local optimum in the first generations. This leads to low population diversity and adaptation stagnation in successive generations.

In order to improve PSO's performance on complex optimization problems, this paper proposed a novel multi-swarm particle swarm optimizer called PS^{2}O, which extend the single population PSO to interacting multi-swarms model by constructing hierarchical interaction topologies and enhanced dynamical update equations. In PS^{2}O, we implement a hierarchical interaction topology that consists of two levels (i.e. individual level and swarm level), in which information exchanges take place permanently. Each individual of the proposed model evolves based on the knowledge integration of itself (associate with individual's own cognition), its swarm members (associate social interaction within each swarm) and its symbiotic partners from other swarm (associate heterogeneous cooperation between different swarms). That is, we extend the control law (i.e. the dynamic update equation) of the canonical PSO model by adding a significant ingredient, which takes into account the symbiotic coevolution (or heterogeneous cooperation) between different swarms. By incorporating this new degree of complexity, PS^{2}O can accommodate a considerable potential for solving more complex problems. Here we provide some initial insights into this potential by evaluating PS^{2}O on both continuous and discrete mathematical benchmark functions. The 17 benchmark functions used in our experiments, which including 10 continuous problems and 7 discrete cases, have been widely employed by other researchers to evaluate their algorithms (Shi and Eberhart 1999; Liang et al. 2006; Clerc 2005). The experimental results, which are compared to other methods, are reported in this paper to show the merits of the proposed algorithm.

The paper is organized as follows. Section 2 gives a review of the canonical PSO algorithm and several multi-swarm PSO variants. Section 3 describes the proposed multi-swarm coevolution algorithm—PS^{2}O, including the hierarchical topology, the enhanced dynamic update formula, and the matrix representation. In Sect. 4, it will be shown that PS^{2}O outperforms the canonical PSO and its variants on 17 benchmark test functions. Finally, conclusions are drawn in Sect. 5.

2 Canonical Particle Swarm Optimization

The canonical PSO is a population-based technique, similar in some respects to evolutionary algorithms, except that potential solutions (particles) move, rather than evolve, through the search space. The rules (or particle dynamics) that govern this movement are

inspired by models of swarming and flocking (Kennedy and Eberhart 2001). Each particle has a position and a velocity, and experiences linear spring-like attractions towards two attractors:

i. Its previous best position.
ii. Best position of its neighbors.

In mathematical terms, the ith particle is represented as $x_i = (x_{i1}, x_{i2}, \ldots x_{iD})$ in the D-dimensional space, where $x_{id} \in [l_d, u_d]$, $d \in [1, D]$, l_d, u_d are the lower and upper bounds for the dth dimension, respectively. The rate of velocity for particle i is represented as $v_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$ is clamped to a maximum velocity $V_{\max}$ which is specified by the user. In each time step t, the particles are manipulated according to the following equations:

$$v_{id}(t) = \chi\big(v_{id}(t-1) + R_1 c_1 (p_{id} - x_{id}(t-1)) + R_2 c_2 \big(p_{gd} - x_{id}(t-1)\big)\big) \tag{1}$$

$$x_{id}(t) = x_{id}(t-1) + v_{id}(t) \tag{2}$$

where R_1 and R_2 are random values between 0 and 1, c_1 and c_2 are learning rates, which control how far a particle will move in a single iteration, p_{id} is the best position found so far of the ith particle, p_{gd} is the best position of any particles in its neighborhood, and χ is called constriction factor (Clerc and Kennedy 2002), given by:

$$\chi = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|} \tag{3}$$

where $\varphi = c_1 + c_2$, $\varphi > 4$.

Kennedy and Eberhart (1997) proposed a binary PSO in which a particle moves in a state space restricted to zero and one on each dimension, in terms of the changes in probabilities that a bit will be in one state or the other. The velocity formula (1) remains unchanged except that x_{id}, p_{id}, and p_{gd} are integers in {0, 1} and v_{id} must be constrained to the interval [0.0, 1.0]. This can be accomplished by introducing a sigmoid function $S(v)$, and the new particle position is calculated using the following rule:

$$\text{if } rand < S(v_{id}),\ \text{then } (x_{id}) = 1;\ \text{else } x_{id} = 0; \tag{4}$$

where *rand* is a random number selected from a uniform distribution in [0.0, 1.0] and the function $S(v)$ is a sigmoid limiting transformation as follows:

$$S(v) = \frac{1}{1 + e^{-v}} \tag{5}$$

We should note that the notion of multi-swarm has been introduced in some variant version of the PSO algorithm. A multi-species PSO (MS-PSO) was proposed by (Chow and Tsui 2004), their species optimizes different function thus realizes the multi-objective optimization. Other variants (Parsopoulos and Vrahatis 2001; Brits et al. 2002) which use species or subswarm have been proposed to encourage the emergence of niches in a single population. However, the niches represent competition rather than cooperation. Frans and Andries used multiple swarms to optimize different components of the solution vector (Van den Bergh and Engelbrech 2004). The approach introduced in (Baskar and Suganthan 2004) relies on having two or more swarms searching concurrently for a solution with frequent interacting, while the interaction frequency is arbitrarily predefined. A multi-swarm cooperative particle swarm optimizer (MCPSO) is described in (Ben et al. 2007). In MCPSO, the population consists of one master swarm and several slave swarms. The slave

swarms execute the canonical PSO algorithm or its variants independently and the particles in the master swarm enhance themselves based on their own knowledge and also the knowledge of the particles in the slave swarms. However, there is no information sharing among slave swarms. According to this limitation, when the slave swarms converge to the local optima, the master swarm will also suffer premature convergence to these local optima in successive generations.

3 PS^{2}O algorithm

Straight PSO uses the analogy of a single-species population and the suitable definition of the particle dynamics and the particle information network (interaction topology) to reflect the social evolution in the population. However, the situation in nature is much more complex than what this simple metaphor seems to suggest. Indeed, in biological populations there is a continuous interplay between individuals of the same species, and also encounters and interactions of various kinds with other species (Tomassini 2005). The points at issue can be clearly seen when one observes such ecological systems as symbiosis, host–parasite systems, and prey–predator systems, in which two organisms mutually support each other, one exploits the other, or they fight against each other. For instance, mutualistic relations between plants and fungi are very common. The fungus invades and lives among the cortex cells of the secondary roots and, in turn, helps the host plant absorb minerals from the soil. Another well-known example is the "association" between the Nile crocodile and the Egyptian plover, a bird that feeds on any leeches attached to the crocodile's gums, thus keeping them clean. This kind of "cleaning symbiosis" is also common in fish.

3.1 The proposed multi-swarm optimizer

Inspired by the mutualism phenomenon, we extend the single population PSO to the interacting multi-swarms model by constructing hierarchical information networks and enhanced particle dynamics. In our multi-swarms approach, the interaction occurs not only between the particles within each swarm but also between different swarms. That is, the information exchanges on a hierarchical topology of two levels (i.e. the individual level and the swarm level). Then, we suggest in the proposed model that each individual moving through the solution space should be influenced by three attractors:

i. Its own previous best position.
ii. Best position of its neighbors from its own swarm.
iii. Best position of its neighbor swarms.

In mathematical terms, our multi-swarm model is defined as a triplet $\langle P, T, C\rangle$, where $P = \{S_1, S_2, \ldots, S_M\}$ is a collection of M swarms, and each swarm possesses a members set $S_k = \left\{X_1^k, X_2^k, \ldots, X_N^k\right\}$ of N individuals. T is the hierarchical topology of the multi-swarm. C is the enhanced control low of the particle dynamics, which can be formulated as:

$$\begin{aligned} v_{id}^k(t) = \chi\Big(v_{id}^k(t-1) + R_1c_1\left(p_{id}^k - x_{id}^k(t-1)\right) + R_2c_2\left(p_{gd}^k - x_{id}^k(t-1)\right) \\ + R_3c_3\left(p_{gd}^\theta - x_{id}^k(t-1)\right)\Big) \end{aligned} \tag{6}$$

$$x_{id}^{k}(t) = x_{id}^{k}(t-1) + v_{id}^{k}(t) \tag{7}$$

where x_{id}^{k} represents the position of the ith particle of the kth swarm, p_{id}^{k} is the personal best position found so far by x_{id}^{k}, p_{gd}^{k} is the best position found so far by this particle's neighbors within swarm k, p_{gd}^{θ} is the best position found so far by the other swarms in the neighborhood of swarm k (here θ is the index of the swarm which the best position belongs to), c_1 is the individual learning rates, c_2 is the social learning rate between particles within each swarm; c_3 is the social learning rate between different swarms and $R_1, R_2, R_3 \in \Re^d$ are random vectors uniformly distributed in [0, 1]. Constriction factor χ is calculated by:

$$\chi = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|} \tag{8}$$

where $\varphi = c_1 + c_2 + c_3$, $\varphi > 4$. Here, the term $R_1 c_1\left(p_{id}^{k} - x_{id}^{k}\right)$ is associated with cognition since it takes into account the individual's own experiences; the term $R_2 c_2\left(p_{gd}^{k} - x_{id}^{k}\right)$ represents the social interaction within swarm k; the term $R_3 c_3\left(p_{gd}^{\theta} - x_{id}^{k}\right)$ takes into account the symbiotic coevolution between dissimilar swarms.

We should note that, for solving discrete problems, we still use Eqs. 4 and 5 to discrete the position vectors in PS^{2}O algorithm.

3.2 Hierarchical interaction topology

In our model, the interaction of agents occurred in a two-level hierarchical topology. Many patterns of connection can be used in different levels of our model. The most common ones are rings, two-dimensional and three-dimensional lattices, stars, and hypercubes (Kennedy and Mendes 2002). In this work, we employed the ring and the star topologies in different levels and obtained four hierarchical interaction topologies, which are illustrated in Fig. 1. In Fig. 1a, b, both levels are structured as stars (or rings). In Fig. 1c, d, four swarms at the upper level are connected by a ring (or a star), while each swarm (possesses four individual particles at the lower level) is structured as a star (or a ring). Namely, the first two topologies are characterized by homogeneous structure (employ the same patterns in both levels) and the other two have the heterogeneous structures (employ different patterns in different levels). Accordingly, four variants of PS^{2}O, namely PS^{2}O-S, PS^{2}O-R, PS^{2}O-S$_R$, and PS^{2}O-R$_S$, each employs a distinct interaction topology of Fig. 1, are comparatively studied in this paper.

3.3 Matrix representation

A multidimensional array representation of the PS^{2}O algorithm is proposed in this section. The PS^{2}O randomly initializes a number of M species (or swarms) with each possesses a number of N particles to represent the biological community in the natural ecosystems. Then the positions X, velocities V and personal best locations P of the community are all specified as the three dimension arrays (showed as in Fig. 2a–c).

Where the first array dimension—*Species number*—is the number of species in the community, the second array dimension—*Swarm size*—is the number of individuals (i.e. particles) in each swarm, and the third dimension—*Dimension*—is the optimization problem dimensionality.

In PS^{2}O model, in order to update the velocity and position arrays, every particle must accelerate to three factors: the previous best position of the individual itself that is called

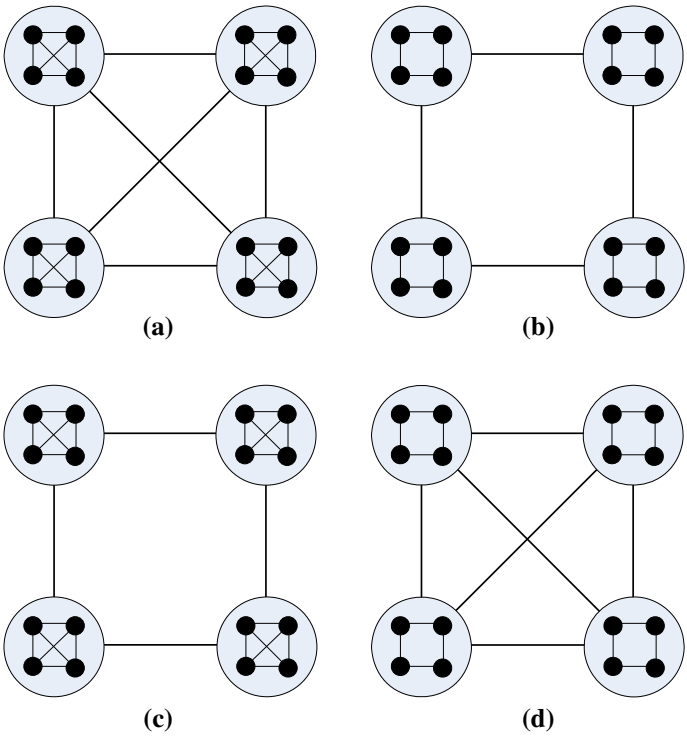

Fig. 1 Hierarchical topologies of the multi-swarm

"personal best" (*pbest*); the previous best position of other species members in its neighborhood and we named this factor "species best" (*sbest*); the previous best position found by other species that have the cooperative relations to the species that this agent belongs to, then we named this factor "community best" (*cbest*). The *sbest* is represented by a matrix S (two dimension array, which showed as in Fig. 2d, left) and the *cbest* is referred to as a one dimension matrix C, which showed as in Fig. 2d, right.

X, V, P, S and C together represent all of the information required by the PS^{2}O algorithm. These arrays are elegantly updated in successive iteration to numerically model the hierarchical emergence. Explicitly, the main difference between PS^{2}O and PSO is the matrix implementation and the modified velocity updating equation, so the complexity of this new algorithm is similar to the original PSO. The pseudocode for the PS^{2}O algorithm is listed in Table 1. The flowchart of the PS^{2}O algorithm is presented in Fig. 3, and according variables used in PS^{2}O are summarized in Table 2.

4 Experiments

4.1 Test function

According to the No Free Lunch theorem, "for any algorithm, any elevated performance over one class of problems is exactly paid for in performance over another class" (Wolpert and Macready 1997). To fully evaluate the performance of the PS^{2}O algorithm without a

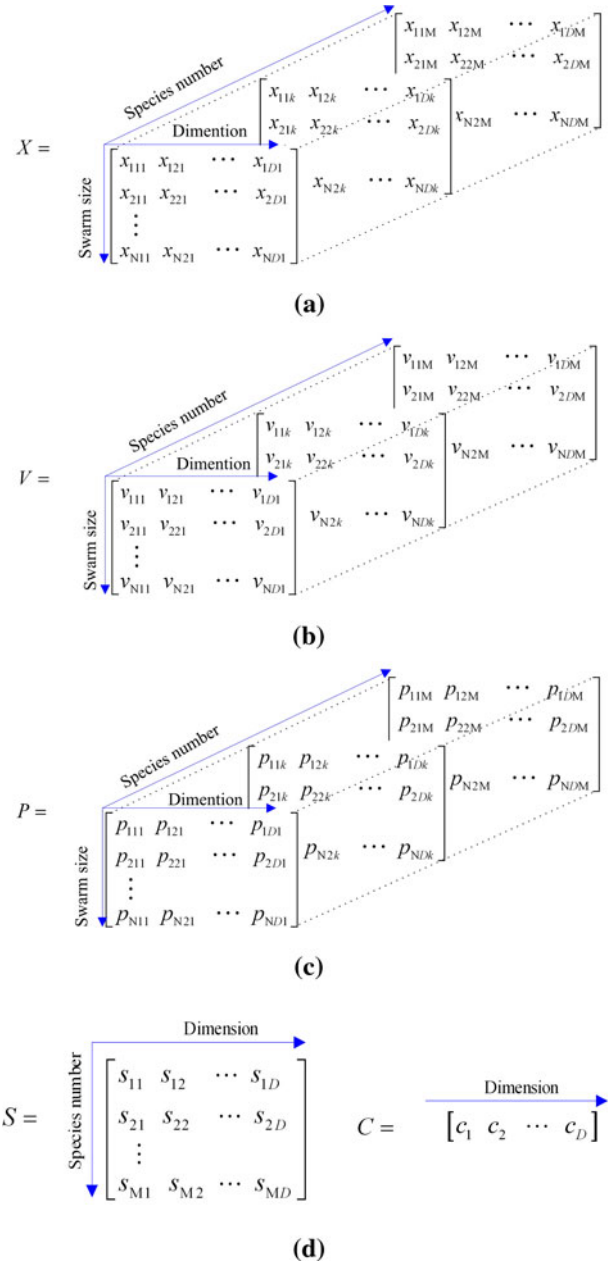

Fig. 2 Matrix representation of the PS^{2}O algorithm

biased conclusion towards some chosen problems, we employed a large set of 17 benchmark functions. These benchmark functions can be grouped as unimodal continuous functions f_1–f_5, multimodal continuous functions f_6–f_{10}, unimodal discrete functions f_{11}–f_{16} and multimodal discrete function f_{17}. Functions f_1–f_9 were widely adopted by other researchers to show solution quality and convergence rate of their algorithms. Function f_{10}

Table 1 Pseudocode for the PS^{2}O algorithm

Set $t := 0$;
INITIALIZE. Randomize n swarms each possesses m particles;
WHILE (the termination conditions are not met)
  FOR (each swarm k)
    Find in the k^{th} swarm neighborhood, the point with the best fitness;
    Set this point as p_{gd}^{θ};
    FOR (each particle i of swarm k)
      Find in the particle neighborhood, the point with the best fitness;
      Set this point as p_{gd}^{k};
      Update particle velocity;
      Update particle position;
    END FOR
  END FOR
  Set $t := t + 1$;
END WHILE

is a novel composition benchmark function developed by Liang (Liang et al. 2005). The discrete functions f_{11}–f_{17} were used by Clerc that formulated in his literature (Clerc 2005) and can be found at http://clerc.maurice.free.fr/pso/. The formulas of these functions are presented below:

1. Sphere function

$$f_1(x) = \sum_{i=1}^{n} x_i^2 \tag{9}$$

2. Rosenbrock function

$$f_2(x) = \sum_{i=1}^{n} 100 \times \left(x_{i+1} - x_i^2\right)^2 + (1 - x_i)^2 \tag{10}$$

3. Quadric function

$$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2 \tag{11}$$

4. Sum of different powers

$$f_4(x) = \sum_{i=1}^{n} |x_i|^{i+1} \tag{12}$$

5. Sin function

$$f_5(x) = \frac{\pi}{n} \left\{ 10 \ \sin^2 \pi x_1 + \sum_{i-1}^{n-1} (x_i - 1)^2 \left(1 + 10 \sin^2 \pi x_{i+1}\right) + (x_n - 1)^2 \right\} \tag{13}$$

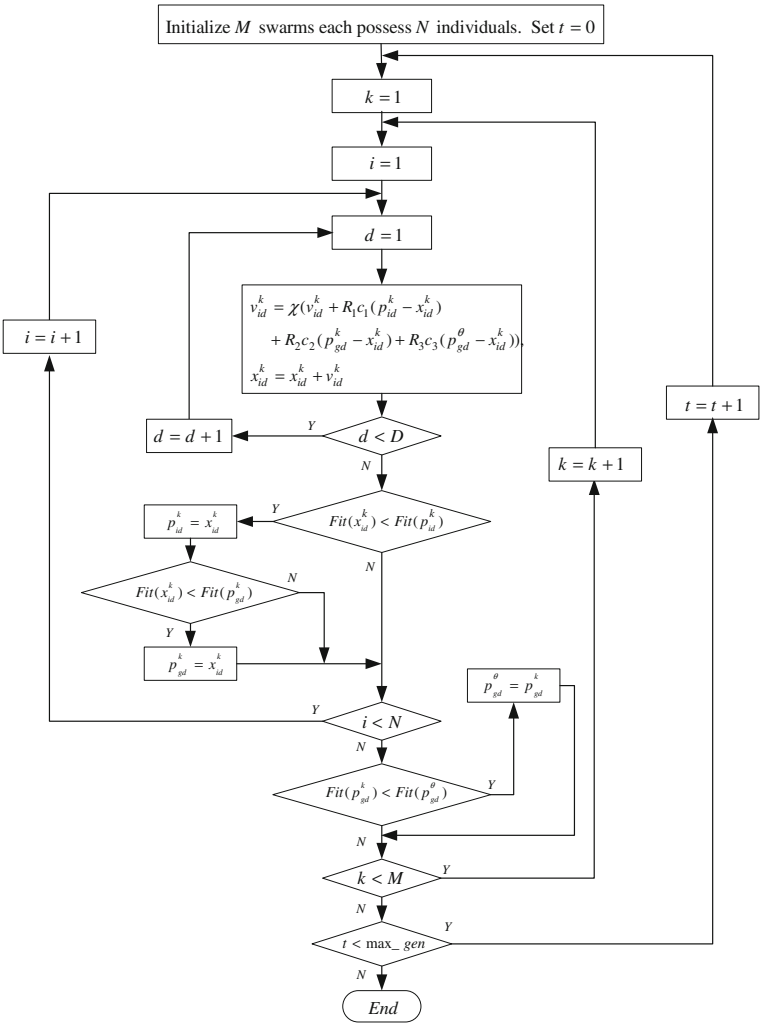

Fig. 3 The flowchart of the PS^{2}O algorithm

6. Ackley's function

$$f_6(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i\right) + 20 + e \tag{14}$$

7. Rastrigrin's function

$$f_7(x) = \sum_{i=1}^{n} x_i^2 - 10\cos(2\pi x_i) + 10 \tag{15}$$

Table 2 List of variables used in PS^{2}O

M	The number of swarms
N	Population size of each swarm
k	Swarm's ID counter from 1 to n
i	Individual's ID counter from 1 to m
d	Dimension of the problem
t	Generation counter from 1 to max generation
θ	The index of the best neighbor swarm of the kth swarm
x_{id}^k	The ith individual's (of the kth swarm) dth dimension's value
p_{id}^k	The ith individual's personal best (of the kth swarm)
p_{gd}^k	The best neighbor position of x_{id}^k in the kth swarm
p_{gd}^θ	The best neighbor position of the kth swarm

8. Griewank function

$$f_8(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \tag{16}$$

9. Weierstrass function

$$f_4(x) = \sum_{i=1}^{n}\left(\sum_{k=0}^{k\max}\left[a^k\cos(2\pi b^k(x_i+0.5))\right]\right) - n\left(\sum_{k=0}^{k\max}\left[a^k\cos(2\pi b^k\cdot 0.5)\right]\right) \tag{17}$$

where $a = 0.5,\ b = 3,\ k\ \max = 20$

10. Composition function 1

The composition function 1 is constructed using 10 unimodal sphere functions. This results in an asymmetrical multimodal function with one global optimum and nine local optima. The variables of the formulation can be referred to (Liang et al. 2005).

$$f_{10}(x) = \sum_{i=1}^{n}\left\{w_i * \left[f_i'((x - o_i + o_{iold})/\lambda_i * M_i) + bias_i\right]\right\} + f\_bias \tag{18}$$

11. Golderg's order-3

The fitness f of a bit-string is the sum of the result of separately applying the following function to consecutive groups of three components each:

$$f_{11}(x) = \begin{cases} 0.9 \text{ if } |y| = 0 \\ 0.6 \text{ if } |y| = 1 \\ 0.3 \text{ if } |y| = 2 \\ 1.0 \text{ if } |y| = 3 \end{cases} \tag{19}$$

If the string size (i.e. the dimension of the problem) is D, the maximum value is $D/3$ for the string 111…111. In practice, we will then use as fitness the value $D/3 - f$ so that the problem is now to find the minimum 0.

12. Bipolar order-6

The fitness f is the sum of the result of applying the following function to consecutive groups of six components each:

$$f_{12}(x) = \begin{cases} 1.0 \text{ if } |y| = 0 \text{ or } 6 \\ 0.0 \text{ if } |y| = 1 \text{ or } 5 \\ 0.4 \text{ if } |y| = 2 \text{ or } 4 \\ 0.8 \text{ if } |y| = 3 \end{cases} \tag{20}$$

The maximum value is $D/6$. In practice, we will use as fitness the value $D/6 - f$ so that the problem is now to find the minimum 0.

13. Mulenbein's order-5

The fitness f is the sum of the result of applying the following function to consecutive groups of five components each:

$$f_{13}(x) = \begin{cases} 4.0 \text{ if } y = 00000 \\ 3.0 \text{ if } y = 00001 \\ 2.0 \text{ if } y = 00011 \\ 1.0 \text{ if } y = 00111 \\ 3.5 \text{ if } y = 11111 \\ 0.0 \text{ otherwise} \end{cases} \tag{21}$$

The maximum value is $3.5D/5$. In practice, the value $3.5D/5 - f$ is use as the fitness so that the problem is now to find the minimum 0.

14. Clerc's Zebra-3

The fitness f is the sum of the result of applying the following function to consecutive groups of three components each, if the rank of the group is even (first rank = 0):

$$f_{14}(x) = \begin{cases} 0.9 \text{ if } |y| = 0 \\ 0.6 \text{ if } |y| = 1 \\ 0.3 \text{ if } |y| = 2 \\ 1.0 \text{ if } |y| = 3 \end{cases} \tag{22}$$

If the rank of the group is odd:

$$f_{14}(x) = \begin{cases} 0.9 \text{ if } |y| = 3 \\ 0.6 \text{ if } |y| = 2 \\ 0.3 \text{ if } |y| = 1 \\ 1.0 \text{ if } |y| = 0 \end{cases} \tag{23}$$

In practice, we will then use as fitness the value $D/3 - f$ so that the problem is now to find the minimum 0.

15. Clerc's order-3 problem 1

The fitness f is the sum of the result of applying the following function to consecutive groups of three components each:

$$f_{15}(x) = \begin{cases} 0.9 \text{ if } |y| = 000 \\ 0.6 \text{ if } |y| = 001 \\ 0.3 \text{ if } |y| = 010 \\ 1.0 \text{ if } |y| = 011 \\ 0.2 \text{ if } |y| = 100 \\ 0.4 \text{ if } |y| = 101 \\ 0.6 \text{ if } |y| = 110 \\ 0.8 \text{ if } |y| = 111 \end{cases} \tag{24}$$

In practice, we will then use as fitness the value $D/3 - f$ so that the problem is now to find the minimum 0.

16. Clerc's order-3 problem 2

The fitness f is the sum of the result of applying the following function to consecutive groups of three components each:

$$f_{16}(x) = \begin{cases} 0.2 \text{ if } |y| = 000 \\ 1.0 \text{ if } |y| = 001 \\ 0.3 \text{ if } |y| = 010 \\ 0.8 \text{ if } |y| = 011 \\ 0.6 \text{ if } |y| = 100 \\ 0.4 \text{ if } |y| = 101 \\ 0.6 \text{ if } |y| = 110 \\ 0.9 \text{ if } |y| = 111 \end{cases} \tag{25}$$

In practice, we will then use as fitness the value $D/3 - f$ so that the problem is now to find the minimum 0.

17. Discrete multimodal problem

This problem's landscape is defined by the following pseudocode:

```
// Definition of the fitness landscape.
For (i = 1 to number of peaks p)
        For (j = 1 to number of Dimension D)
            landscape (i, j) = rand();
        End for
End for
// the fitness f is computed as follows:
f = 0;
For (i = 1 to number of peaks p)
        q = 0;
        For (j = 1 to number of Dimension D)
            If (x(j) = landscape (i, j))
                q = q + 1;
            End if
```

End for
If ($f < q$)
$f = q$;
End if
End for
$f = 1 - f/D$.

The dimensions, initialization ranges, global optima x^*, and the corresponding fitness value $f(x^*)$ of each function are listed in Table 3.

4.2 Experimental setting

Experiments were conducted with four variant versions of (PS^{2}Os), namely PS^{2}O-S, PS^{2}O-R, PS^{2}O-S$_R$, and PS^{2}O-R$_S$ based on four different hierarchical interaction topologies. The following four successful PSO algorithms were also executed for comparisons:

- Local version of PSO with constriction factor (CPSO) (Kennedy and Mendes 2002);
- Fully informed particle swarm (FIPS) (Mendes et al. 2004);
- Unified particle swarm (UPSO) (Parsopoulos and Vrahatis 2004);
- Fitness-Distance-Ratio based PSO (FDR-PSO) (Veeramachaneni et al. 2003).

Among these variations, UPSO combined the global version and local version PSO together to construct a unified particle swarm optimizer; FIPS used all the neighbors' knowledge of the particle to update the velocity; when updating each velocity dimension,

Table 3 Parameters of the test functions

	Dimensions	Initial range	x^*	$f(x^*)$
f_1	30	$[-100, 100]^D$	[0,0,…,0]	0
f_2	30	$[-30, 30]^D$	[1,1,…,1]	0
f_3	30	$[-65.536, 65.536]^D$	[0,0,…,0]	0
f_4	30	$[-1, 1]^D$	[0,0,…,0]	0
f_5	30	$[-10, 10]^D$	[0,0,…,0]	0
f_6	30	$[-32.768, 32.768]^D$	[0,0,…,0]	0
f_7	30	$[-5.12, 5.12]^D$	[0,0,…,0]	0
f_8	30	$[-600, 600]^D$	[0,0,…,0]	0
f_9	30	$[-0.5, 0.5]^D$	[0,0,…,0]	0
f_{10}	10	$[-5, 5]^D$	Predefined rand number distributed in the search range	0
f_{11}	120	[0, 1]	[1,1,…,1]	0
f_{12}	120	[0, 1]	[0,0,…,0] or [1,1,…,1] or […,6*0,…,6*1,…]	0
f_{13}	120	[0, 1]	[0,0,…,0]	0
f_{14}	30	[0, 1]	[0,0,0,1,1,1,0,0,0,1,1,1,…]	0
f_{15}	60	[0, 1]	[0,1,1,0,1,1,…]	0
f_{16}	60	[0, 1]	[0,0,1,0,0,1,…]	0
f_{17}	200	[0, 1]	Predefined rand number distributed in the search range	0

the FDR-PSO selects one other particle, which has a higher fitness value and is nearer to the particle being updated.

The population size of all algorithms tested in this experiment was set at 100. For continuous problems, the maximum velocity for all algorithms was set to be 5% of the search space for unimodal continuous functions and 50% for multimodal continuous functions. For canonical PSO, UPSO and FIPS, the learning rates c_1 and c_2 were both 2.05 and the constriction factor $\chi = 0.729$. For FDR-PSO, the inertia weight ω started at 0.9 and ended at 0.5 and a setting of $c_1 = c_2 = 2.0$ was adopted. For each PS^{2}O algorithm, except different interaction topologies are used, the parameters were set to the values $c_1 = c_2 = c_3 = 1.3667$ and $\chi = 0.729$. Since there are no literatures using UPSO, FIPS and FDR-PSO for discrete optimization so far, we just test four PS^{2}O variants compared with the binary version PSO on discrete problems. For discrete problems, the maximum velocity for all algorithms was set to be 4. The constriction factor $\chi = 1$ for both PS^{2}Os and PSO. The learning rates were set to the values $c_1 = c_2 = c_3 = 2.0$ and $c_1 = c_2 = 2.0$ for PS^{2}Os and PSO respectively.

The number of swarms M needs be tuned. Six benchmark functions—Sphere 10D, Rosenbrock 10D, Rastrigrin 10D, Golderg 120D, Bipolar 60D, and Discrete multimodal problem 100D—are used to investigate the impact of this parameter. Experiments were executed with PS^{2}O-R on Sphere, PS^{2}O-S$_R$ on Rosenbrock, PS^{2}O-S on Rastrigrin, PS^{2}O-R on Golderg, PS^{2}O-R$_S$ on Bipolar, and PS^{2}O-S on Discrete multimodal problem by changing the number of swarms and fixing each swarm size at 10. The average test results obtained from 30 runs are plot in Fig. 4. For continuous problems, the performance measurement is the average best-so-far fitness value; while for discrete cases, the performance measurement is the mean iteration to the function minimum 0. From Fig. 4, we can observe that the performance of PS^{2}O is influenced by M. When M increases, we obtained faster convergence velocity and better results. In following experiments, M is set at 10 for all PS^{2}Os (i.e. each swarm possesses $N = 100/10 = 10$ particles).

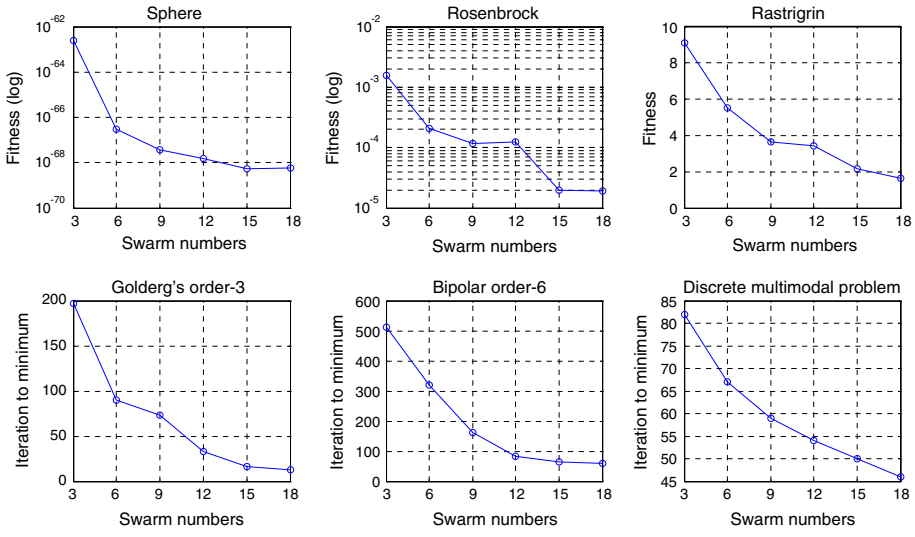

Fig. 4 PS^{2}Os' results on six test functions with different number of swarm M

The experiment runs 30 times respectively for each algorithm on each benchmark function. The number of generations for the 10 continuous benchmark functions was set to be 10,000 and for the seven discrete functions was 1000.

4.3 Continuous unimodal functions

Unimodal problems have been adopted to assess the convergence rates of optimization algorithms. We test the four PS^{2}O variants on a set of unimodal functions in comparison with PSO, FIPS, UPSO and FDR-PSO algorithms. Table 4 lists the experimental results (i.e., the mean and standard deviations of the function values found in 30 runs) for each algorithm on unimodal functions f_1–f_5. Figure 5 shows the search progress of the average values found by the eight algorithms over 30 runs for functions f_1–f_5.

From Table 4 and Fig. 5, the four PS^{2}O variants converged much faster to significantly better results than all other algorithms. The PS^{2}O-R$_S$ that has the heterogeneous hierarchical structures is the fastest one for finding good results within relatively few generations. All PS^{2}O variants were able to consistently find the minimum to functions f_1, f_4 and f_5 within 10,000 generations. When applying the PS^{2}O-R$_S$ to Rosenborck function f_2, the mean function values consistently reach the remarkable good value of 3.71e−026. The results on Rosenbrock obtained by PS^{2}O-R and PS^{2}O-S$_R$ are also very good.

From the comparisons between PS^{2}Os and other algorithms, we can see that, statistically, PS^{2}Os have significantly better performance on continuous unimodal functions f_1–f_5. From the rank values presented in Table 4, the search performance of the algorithms tested here is ordered as PS^{2}O-S > PS^{2}O-R > PS^{2}O-S$_R$ > PS^{2}O-R$_S$ > UPSO > FDR-PSO > PSO > FIPS.

4.4 Continuous multimodal functions

The first four multi-modal functions f_6–f_9 are regarded as the most difficult functions to optimize since the number of local minima increases exponentially as the function dimension increases. According to the results reported in (Liang et al. 2005, 2006), the methods CL-PSO, PSO, CMA-ES, G3-PCX, DE and other algorithms used for comparison all failed to find the minimum of six composition function. Since these mentioned methods have demonstrated their excellent performance on standard benchmark functions, the six composition functions are very complex. In this paper, we only test PS^{2}Os on the first composition function f_{10} and the test on the other five composition functions will be studied in the future works. The mean and standard deviations of the function values found in 30 runs for each algorithm on each function are listed in Table 5. Figure 6 shows the search progress of the average values found by the eight algorithms over 30 runs for functions f_6–f_{10}.

From Table 5 and Fig. 6, it is clear to see that PS^{2}O algorithms outperformed the other four algorithms. For example, PS^{2}O-R and PS^{2}O-S$_R$ found the global minimum every time of run on function f_8–f_{10} and PS^{2}O-S can also consistently found the minimum of f_{10} within relatively fewer generations, while the other algorithms generated poorer results on them. On function f_6 and f_7, the four PS^{2}O algorithms yielded similar results to the other four algorithms. From the rank values presented Table 5, the search performance of the algorithms tested here is ordered as PS^{2}O-R > PS^{2}O-S$_R$ > PS^{2}O-S > FIPS > FDR-PSO > PSO > UPSO > PS^{2}O-R$_S$.

It should be mentioned that PS^{2}O were the only ones able to consistently find the minimum to the composition function F1 that reported in the literatures so far.

Table 4 Performance of all algorithms on benchmark functions f_1–f_5

Func.	PS^{2}O-S	PS^{2}O-R	PS^{2}O-S$_R$	PS^{2}O-R$_S$	PSO	FIPS	UPSO	FDR-PSO
f_1								
Mean	**0**	**0**	6.24e−305	**0**	2.42e−114	1.73e−029	3.71e−183	2.47e−169
Std	**0**	**0**	**0**	**0**	3.39e−114	2.29e−029	**0**	**0**
Rank	**1**	**1**	4	**1**	7	8	6	5
f_2								
Mean	1.32e−001	1.04e−014	2.55e−016	**3.71e−026**	6.61e+000	2.25e+001	2.09e+000	2.79e−001
Std	7.27e−001	9.71e−015	1.70e−016	**1.09e−025**	4.02e−001	1.27e+000	7.69e−001	1.02e+000
Rank	4	3	2	**1**	7	8	6	5
f_3								
Mean	1.04e−090	2.19e−047	1.42e−051	**2.50e−092**	3.51e−018	1.38e+000	6.12e−013	4.25e−014
Std	5.01e−090	6.82e−047	4.28e−051	**1.02e−091**	7.88e−018	1.00e+000	6.06e−013	2.32e−013
Rank	2	4	3	**1**	5	8	7	6
f_4								
Mean	**0**	**0**	**0**	8.23e−320	4.54e−185	5.03e−065	**0**	8.64e−280
Std	**0**	**0**	**0**	**0**	**0**	2.38e−064	**0**	**0**
Rank	**1**	**1**	**1**	**5**	**7**	8	**1**	**6**
f_5								
Mean	**0**	**0**	1.38e−002	1.14e−001	4.84e−002	1.38e−002	**0**	3.50e−003
Std	**0**	**0**	3.58e−002	2.80e−001	8.04e−002	4.50e−002	**0**	1.89e−002
Rank	**1**	**1**	5	8	7	6	**1**	4
Aver. rank	**1.8**	2	3	3.2	6.6	7.6	4.2	5.2
Final rank	**1**	2	3	4	7	8	5	6

In bold are the best results

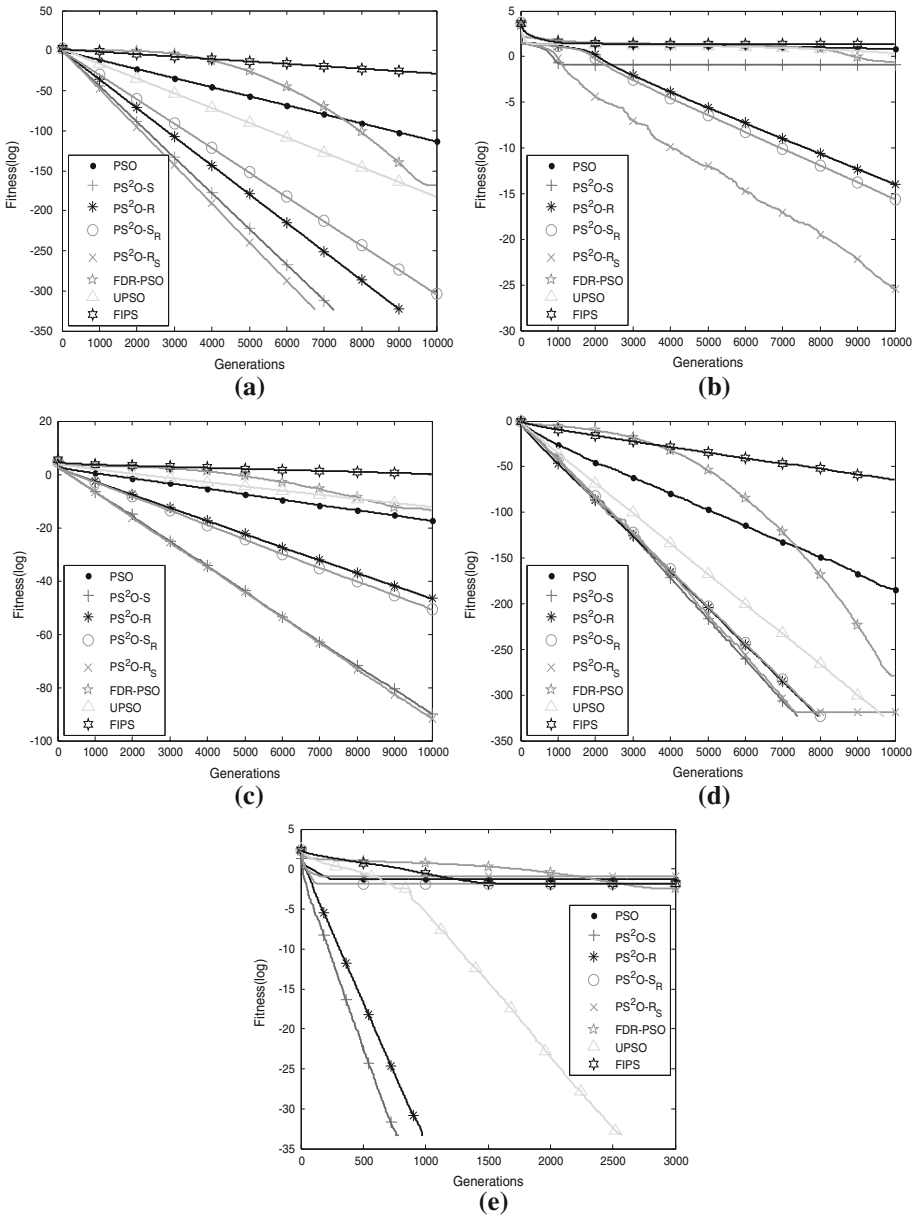

Fig. 5 The median convergence results of 30D unimodal continuous functions. **a** f_1, **b** f_2, **c** f_3, **d** f_4, **e** f_5

4.5 Discrete functions

In binary optimization, it is very easy to design some algorithms that are extremely good on some benchmarks, while extremely bad on some others (Clerc 2005). In order to fully evaluate the performance of PS^{2}Os on discrete problems, we have employed a carefully chosen set of discrete benchmark functions f_{11}–f_{17}. The results obtained by four PS^{2}O

Table 5 Performance of all algorithms on benchmark fucions f_6–f_{10}

Func.	PS^{2}O-S	PS^{2}O-R	PS^{2}O-S$_R$	PS^{2}O-R$_S$	PSO	FIPS	UPSO	FDR-PSO
f_6								
Mean	7.11e−015	4.74e−015	4.97e−015	8.32e−002	6.16e−015	2.04e−014	**3.55e−015**	2.74e−014
Std	**0**	1.70e−015	1.77e−015	3.17e−002	1.80e−015	1.08e−014	3.16e−015	1.03e−014
Rank	**1**	3	4	8	5	6	2	7
f_7								
Mean	4.47e+001	3.25e+001	3.61e+001	4.61e+001	2.78e+001	4.13e+001	4.96e+001	**2.60e+001**
Std	1.20e+001	8.41e+000	8.96e+000	1.11e+001	**5.46e+000**	2.34e+001	9.80e+000	8.32e+000
Rank	7	3	4	6	2	5	8	**1**
f_8								
Mean	8.37e−003	**0**	**0**	9.26e−003	1.83e−002	1.56e−003	3.47e−003	1.79e−002
Std	1.09e−002	**0**	**0**	1.07e−002	2.66e−002	3.78e−003	4.78e−003	1.82e−002
Rank	5	**1**	**1**	6	8	3	4	7
f_9								
Mean	1.17e+000	**0**	**0**	1.22e+000	1.35e+000	2.01e−002	4.42e+000	1.58e−001
Std	7.68e−001	**0**	**0**	8.64e−001	1.16e+000	5.58e−002	2.60e+000	4.56e−001
Rank	5	**1**	**1**	6	7	3	8	4
f_{10}								
Mean	**0**	**0**	**0**	4.00e+001	5.00e+001	3.37e+001	8.00e+001	1.00e+002
Std	**0**	**0**	**0**	8.94e+000	1.82e+002	2.58e+001	4.42e+001	4.42e+001
Rank	**1**	**1**	**1**	5	6	4	7	8
Aver. rank	**3.8**	**1.8**	**3**	6.2	5.6	4.2	5.8	5.4
Final rank	**3**	**1**	**2**	8	6	4	7	5

In bold are the best results

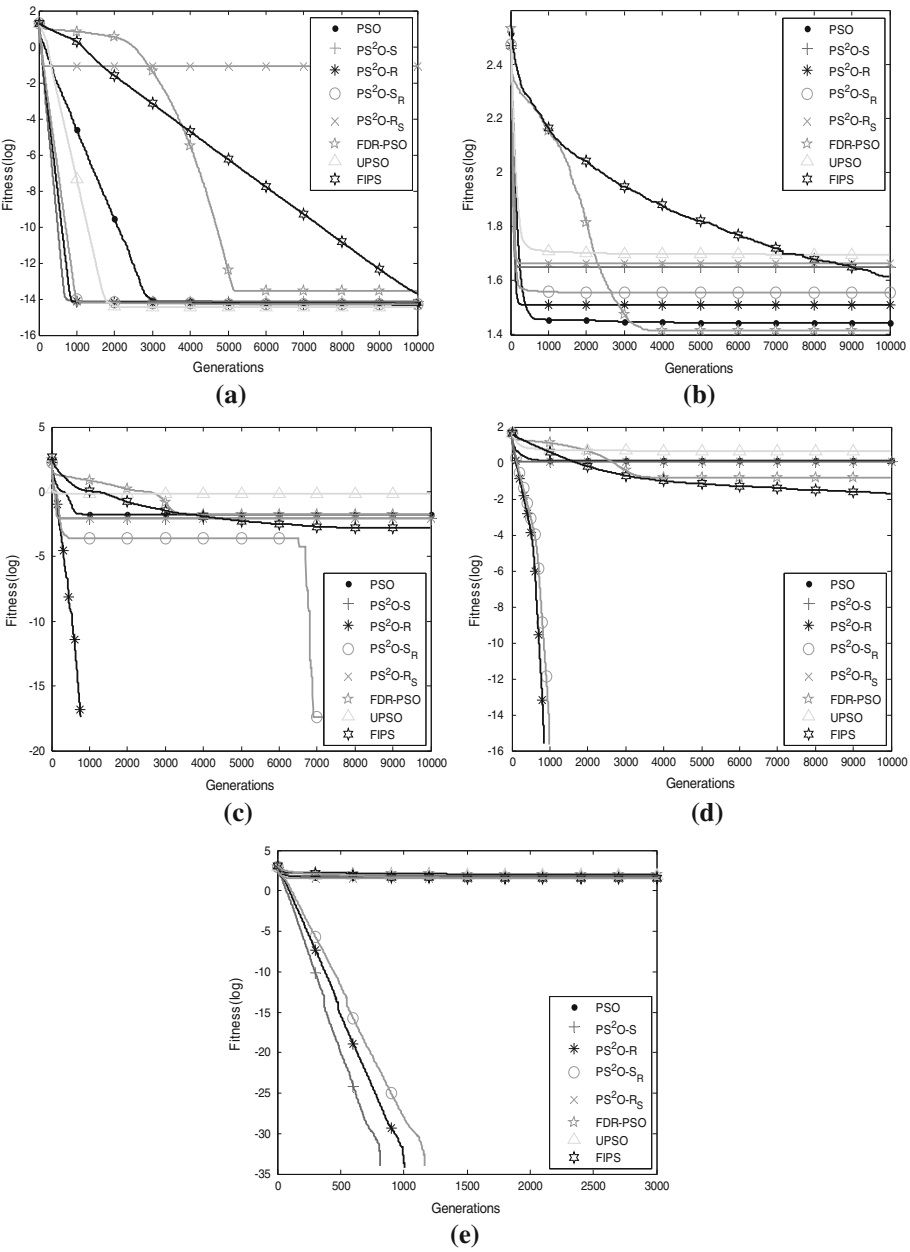

Fig. 6 The median convergence results of 30D multimodal continuous functions. **a** f_6, **b** f_7, **c** f_8, **d** f_9, **e** f_{10}

algorithms and canonical PSO on each discrete benchmark function are listed in Table 6, including the mean number of iterations require to reach the minimum 0, mean and standard deviations of the function values found in 30 runs. Figure 7 shows the search progress of the average values found by the five algorithms over 30 runs for functions f_{11}–f_{17}.

Table 6 Performance of all algorithms on benchmark functions f_{11}–f_{17}

Func.	PS^{2}O-S	PS^{2}O-R	PS^{2}O-S$_R$	PS^{2}O-R$_S$	PSO
f_{11}					
Mean	**0**	**0**	1.67e−002	**0**	2.14e+000
Std	**0**	**0**	5.31e−002	**0**	2.60e−001
Iter.	**14**	24	335	28	∞
Rank	**1**	2	4	3	5
f_{12}					
Mean	**0**	**0**	3.33e−002	**0**	3.19e+000
Std	**0**	**0**	1.18e−001	**0**	3.81e−001
Iter.	310	**69**	493	328	∞
Rank	2	**1**	4	3	5
f_{13}					
Mean	**0**	**0**	**0**	**0**	3.96e+000
Std	**0**	**0**	**0**	**0**	1.33e+000
Iter.	134	**82**	426	275	∞
Rank	2	**1**	4	3	5
f_{14}					
Mean	4.00e−003	**0**	6.70e−003	**0**	2.94e−001
Std	1.98e−002	**0**	2.54e−002	**0**	1.33e−002
Iter.	871	**367**	538	398	∞
Rank	4	**1**	3	2	5
f_{15}					
Mean	**0**	**0**	**0**	**0**	**0**
Std	**0**	**0**	**0**	**0**	**0**
Iter.	**17**	89	147	107	356
Rank	**1**	2	4	3	5
f_{16}					
Mean	**0**	**0**	**0**	**0**	2.00e−003
Std	**0**	**0**	**0**	**0**	1.41e−002
Iter.	203	272	187	**151**	436
Rank	3	4	2	**1**	5
f_{17}					
Mean	**0**	**0**	**0**	**0**	**0**
Std	**0**	**0**	**0**	**0**	**0**
Iter.	**39**	51	**39**	55	61
Rank	**1**	3	**1**	4	5
Aver. rank	**2**	**2**	3.14	2.71	5
Final rank	2	**1**	4	3	5

In bold are the best results

From the results, we can observe that PS^{2}Os obtain an obviously remarkable performance. It can be seen from Fig. 7 that all PS^{2}O variants converged greatly faster and to significantly better results—the minimum of functions f_{11}–f_{17}—than PSO for all discrete cases.

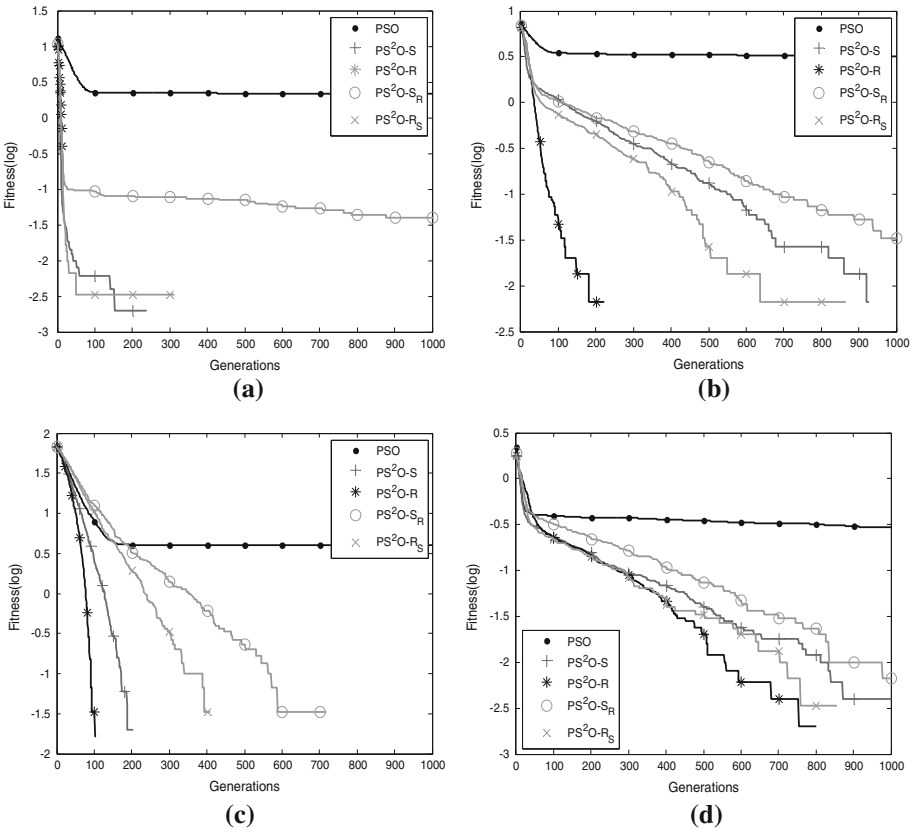

Fig. 7 The median convergence results of discrete functions. **a** f_{11}, **b** f_{12}, **c** f_{13}, **d** f_{14}, **e** f_{15}, **f** f_{16}, **g** f_{17}

It can be found from Table 6 that, the PS^{2}O-R was ranked No. 1 with the best results and robust performance. From the rank values presented in Table 6, the search performance of the algorithms tested here is ordered as PS^{2}O-R > PS^{2}O-S > PS^{2}O-R$_S$ > PS^{2}O-S$_R$ > PSO. It is worth mentioning that the PS^{2}O-R and PS^{2}O-R$_S$ were able to consistently find the minimum to all discrete benchmark functions.

5 Conclusions

Inspired by the biological coevolution phenomenon between different species, a multi-swarm coevolution optimizer (called PS^{2}O) has been proposed to improve the performance of original PSO. PS^{2}O extends the single population PSO to interacting multi-swarms model by constructing hierarchical interaction topologies and enhanced dynamical update equations. In PS^{2}O, the hierarchical interaction topology consists of two levels (i.e. the individual level and the swarm level), in which information exchanges take place not only between the particles within each swarm but also between different swarms. The dynamical update equations of our multi-swarm approach are enhanced by a significant ingredient, which takes into account the symbiotic coevolution (or heterogeneous cooperation) between different swarms. Because of this, each individual of the proposed model

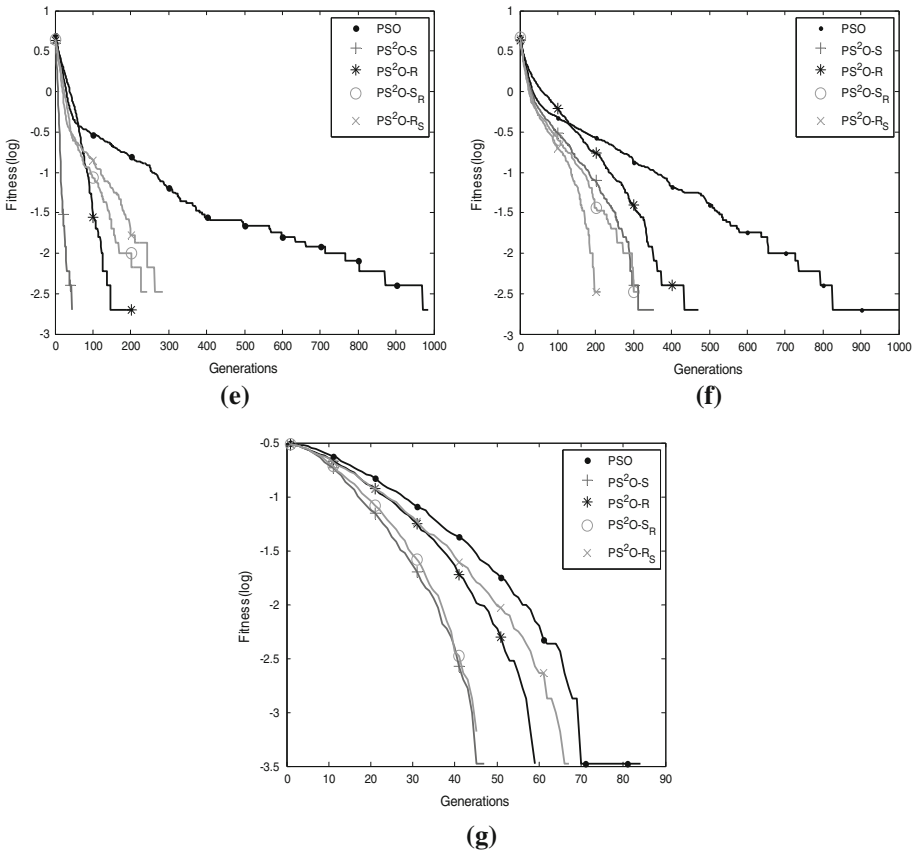

Fig. 7 continued

evolves based on the knowledge integration of itself (associate with individual's own cognition), its swarm members (associate social interaction within each swarm) and its symbiotic partners from other swarm (associate heterogeneous cooperation between different swarms).

With the hierarchical interaction topology, a suitable diversity in the whole population can be maintained. At the same time, the enhanced dynamical update rule significantly speeds up the multi-swarm to converge to the global optimum. This is proven by the comparison with four variant versions of PSO on 17 benchmark functions including both continuous and discrete cases. The experiment results show that, for all the test functions, the PS^{2}O reaches remarkably better results than the other four PSO variants.

There are ways to improve our proposed PS^{2}O algorithm. Further research efforts should focus on:

(1) In PS^{2}O model, only mutualistic relations between different species are mimicked. More symbiosis types could be studied and to be incorporate in PS^{2}O, such as commensalisms, and parasitism.
(2) Moreover, it remains to be see how practically useful the PS^{2}O algorithm are for real-world complex engineering optimization problems.

Acknowledgements This work is supported by the National 863 plans projects of China under Grant 2006AA04A117 and the National 863 plans projects of China under Grant 08H2010201. The authors would like to thank anonymous reviewers for their valuable suggestions.

References

Angeline PJ (1998) Evolutionary optimization versus particle swarm optimization: philosophy and performance difference. In: Proceedings of the 7th annual conference on evolutionary programming, San Diego, USA, pp 601–610

Bäck T, Schwefel HP (1995) Evolution strategies I: variants and their computational implementation. Genetic algorithms in engineering and computer science. Wiley, Chichester, pp 111–126

Baskar S, Suganthan PN (2004) A novel concurrent particle swarm optimization. In: Proceedings of the 2004 congress on evolutionary computation, vol 1, pp 792–796

Ben N, Yunlong Z, XiaoXian H, Xiangping Z, Henry W (2007) MCPSO: a multi-swarm cooperative particle swarm optimizer. Appl Math Comput 185(2):1050–1062

Brits R, Engelbrecht AP, Bergh F (2002) A niching particle swarm optimizer. In: Proceedings of the 4th Asia-Pacific conference on simulated evolution and learning, pp 692–696

Chow CK, Tsui HT (2004) Autonomous agent response learning by a multi-species particle swarm optimization. In: Proceeding of congress on evolutionary computation, Portland, Oregon, USA, pp 778–785

Clerc M (2005) Binary particle swarm optimizers: toolbox, derivations, and mathematical insights. Available: http://clerc.maurice.free.fr/pso/

Clerc M, Kennedy J (2002) The particle swarm-explosion, stability, and convergence in a multidimensional complex space. IEEE Trans Evol Comput 6(1):58–73

Dorigo M, Gambardella LM (1997) Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans Evol Comput 1:53–66

Eberhart RC, Kennedy J (1995) A new optimizer using particle swarm theory. In: Proceeding of the 6th international symposium on micromachine and human science, Nagoya, Japan, pp 39–43

Holland JH (1975) Adaptation in natural and artificial systems. The University of Michigan Press, Ann Arbor

Karaboga D, Basturk B (2008) On the performance of artificial bee colony (ABC) algorithm. Appl Soft Comput 8(1):687–697

Kennedy J, Eberhart RC (1997) A discrete binary version of the particle swarm algorithm. In: Proceeding of the world multiconference on systemics, cybernetics and informatics, Piscataway, pp 4104–4109

Kennedy J, Eberhart RC (2001) Swarm intelligence. Morgan Kaufmann, San Francisco

Kennedy J, Mendes R (2002) Population structure and particle swarm performance. In: Proceeding of the congress on evolutionary computation, Honolulu, HI, pp 1671–1676

Koza JR (1992) Genetic programming: on the programming of computers by means of natural selection. MIT Press, Cambridge

Liang JJ, Suganthan PN, Deb K (2005) Novel composition test functions for numerical global optimization. In: Proceeding of the 2005 swarm intelligence symposium, pp 68–75

Liang JJ, Suganthan PN, Qin AK, Baskar S (2006) Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Trans Evol Comput 10(3):281–295

Mendes R, Kennedy J, Neves J (2004) The fully informed particle swarm: simpler, maybe better. IEEE Trans Evol Comput 8(3):204–210

Parsopoulos KE, Vrahatis MN (2001) Modification of the particle swarm optimizer for locating all the global minima. In: Proceedings of the international conference on artificial neural networks and genetic algorithms, Prague, Czech Republic, pp 324–327

Parsopoulos KE, Vrahatis MN (2004) UPSO: a unified particle swarm optimization scheme. LNCS, pp 868–873

Passino KM (2002) Biomimicry of bacterial foraging for distributed optimization and control. IEEE Control Syst Mag 22:52–67

Shi Y, Eberhart RC (1999) Empirical study of particle swarm optimization. In: Proceeding of the 1999 IEEE congress on evolutionary computation, Piscataway, NJ, pp 1945–1950

Tomassini M (2005) Spatially structured evolutionary algorithms: artificial evolution in space and time. Springer

Van den Bergh F, Engelbrech AP (2004) A cooperative approach to particle swarm optimization. IEEE Trans Evol Comput 8(3):225–239

Veeramachaneni K, Peram T, Mohan CK, Osadciw LA (2003) Optimization using particle swarms with near neighbor interactions. In: Proceeding of the genetic and evolutionary computation conference, Chicago, IL, USA, pp 110–121

Wolpert DH, Macready WG (1997) No free lunch theorems for search. IEEE Trans Evol Comput 1(1): 67–82

Yao X, Liu Y, Lin G (1999) Evolutionary programming made faster. IEEE Trans Evol Comput 3(2):82–102