Available online at www.sciencedirect.com
ScienceDirect
Applied Mathematics and Computation 185 (2007) 1050–1062
APPLIED MATHEMATICS AND COMPUTATION
www.elsevier.com/locate/amc

MCPSO: A multi-swarm cooperative particle swarm optimizer

Ben Niu [a,b,*], Yunlong Zhu [a], Xiaoxian He [a,b], Henry Wu [c]

[a] *Shenyang Institute of Automation, Chinese Academy of Sciences, Faculty Office II, Nanta Street 114#, Donling District, Shenyang 110016, China*
[b] *Graduate School of the Chinese Academy of Sciences, Beijing 100049, China*
[c] *Department of Electrical Engineering and Electronics, The University of Liverpool, Liverpool L69 3GJ, UK*

Abstract

This paper presents a new optimization algorithm – MCPSO, multi-swarm cooperative particle swarm optimizer, inspired by the phenomenon of symbiosis in natural ecosystems. MCPSO is based on a master–slave model, in which a population consists of one master swarm and several slave swarms. The slave swarms execute a single PSO or its variants independently to maintain the diversity of particles, while the master swarm evolves based on its own knowledge and also the knowledge of the slave swarms. According to the co-evolutionary relationship between master swarm and slave swarms, two versions of MCPSO are proposed, namely the competitive version of MCPSO (COM-MCPSO) and the collaborative version of MCPSO (COL-MCPSO), where the master swarm enhances its particles based on an antagonistic scenario or a synergistic scenario, respectively. In the simulation studies, several benchmark functions are performed, and the performances of the proposed algorithms are compared with the standard PSO (SPSO) and its variants to demonstrate the superiority of MCPSO.

Keywords: Particle swarm optimization; Multi-swarm; Master–slave model; MCPSO

1. Introduction

Particle swarm optimization (PSO) was motivated from the simulation of simplified social behavior of animals, firstly developed by Kennedy and Eberhart [1,2]. Compared to other evolutionary algorithms, such as genetic algorithm (GA), the advantages of PSO are that PSO is easy in implementation and there are few parameters to adjust. It has already been applied successfully in many application areas, including function optimization [3,4], artificial neural network training [5,6], fuzzy system control [7,8], optimizing power flow [9], NC programming [10] and other areas where GA can be applied to.

However, the studies by Angeline [11] showed that the original PSO (or SPSO) had difficulties in controlling the balance between exploration (global investigation of the search place) and exploitation (the fine search

* Corresponding author. Address: Shenyang Institute of Automation, Chinese Academy of Sciences, Faculty Office II, Nanta Street 114#, Donling District, Shenyang 110016, China.
E-mail address: niuben@sia.cn (B. Niu).

0096-3003/$ - see front matter
doi:10.1016/j.amc.2006.07.026

around a local optimum). The SPSO, while quickly converging towards an optimum in the first period of iterations, has problems when it comes to reach a near optimal solution.

Various attempts have been made to improve the performance of SPSO, including such changes as hybrid models [12,13], biology-inspired mechanisms [14,15], and of course tunable parameters in the velocity update equations of SPSO [4].

We present here a new approach to balance the exploration and exploitation in PSO, by introducing a multi-swarm cooperative scheme, which consists of one master swarm and several slave swarms. The slave swarms can supply many new promising particles (the position giving the best fitness value) to the master swarm as the evolution proceeds. The master swarm updates the particle states based on the best position discovered so far by all the particles both in the slave swarms and its own. The interactions between the master swarm and slave swarms influence the balance between exploration and exploitation and maintain a suitable diversity in the population, even when it is approaching the global solution, thus reducing the risk of converging to local sub-optima.

According to the co-evolutionary relationship between master swarm and slave swarms, two new MCPSO models, the competitive version of MCPSO (COM-MCPSO) and the collaborative version of MCPSO (COL-MCPSO), are proposed. COM-MCPSO is based on an antagonistic scenario, where the master swarm enhances its particles by a series of competitions with the slave warms. In contrast, COL-MCPSO follows a synergistic scenario, where the master swarm updates its particles by a series of collaborations with the slave warms. Several well-know benchmark functions are chosen to compare the MCPSO algorithms with SPSO and its variants.

The rest of the paper is organized as follows. Section 2 describes the SPSO along with its variants. Section 3 motivates and describes the MCPSO algorithm and gives the pseudo code of the algorithm. Section 4 defines the benchmark problems used for experiment comparison of the algorithms, and the experimental settings for each algorithm. Section 5 presents the results followed by conclusions in Section 6.

2. Review of standard PSO

The fundament for the development of PSO is hypothesis that a potential solution to an optimization problem is treated as a bird without quality and volume, which is called a particle, flying through a D-dimensional space, adjusting its position in search space according to its own experience and that of its neighbors.

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 of the dth dimension, respectively. The velocity for particle i is represented as $v_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$, which is clamped to a maximum velocity $V_{\max}$, specified by the user. In each time step t, the particles are manipulated according to the following equations:

$$v_i(t+1) = v_i(t) + R_1 c_1 (P_i - x_i(t)) + R_2 c_2 (P_g - x_i(t)), \tag{1}$$

$$x_i(t+1) = x_i(t) + v_i(t), \tag{2}$$

where R_1 and R_2 are random values between 0 and 1. c_1 and c_2 are acceleration constants, which control how far a particle will move in a single iteration. P_i is the best previous position of the ith particle. According to the different definitions of P_g there are two different versions of PSO. If P_g is the best position among all the particles in the swarm (also known as *gbest*), such a version is called the global version. If P_g is taken from some smaller number of adjacent particles of the population (also known as *lbest*), such a version is called the local version.

Shi and Eberhart [4] later introduced an inertia term w by modifying (1) to

$$v_i(t+1) = w \times v_i(t) + R_1 c_1 (P_i - x_i(t)) + R_2 c_2 (P_g - x_i(t)). \tag{3}$$

They proposed that suitable selection of w will provide a balance between global and local explorations, thus requiring less iterations on average to find a sufficiently optimal solution. As originally developed, w often decreases linearly from about 0.9 to 0.4 during a run. In general, the inertia weight w is set according to the following equation:

$$w = w_{\max} - \frac{w_{\max} - w_{\min}}{\text{iter}_{\max}} \times \text{iter}, \tag{4}$$

where w_{max} and w_{min} are the initial weight and final weight, respectively, $iter_{max}$ is the maximum number of allowable iterations, and iter the current iteration number. Hereafter, in this paper, this version of PSO is referred to as a linearly decrease inertia weight method (LPSO).

Aside from LPSO, Eberhart and Shi [16] have also proposed a random inertia weight factor for tracking dynamic systems. In this development, the inertia weight factor is set to change randomly according to the following equation:

$$w = 0.5 - \frac{\text{rand}()}{2}, \tag{5}$$

where rand( ) is a uniformly distributed random number within range $[0, 1]$. It is recommended that the acceleration coefficients keep constant at 1.494. In the remainder of this paper, this method is referred to as random weight method (RPSO).

Another important modification of SPSO is the constriction factor approach PSO (CPSO), which was proposed by Clerc and Kenedy [17]. A detailed discussion of the constriction factor is beyond the scope of this paper, but a simplified method of incorporating it described in Eq. (6), where K is a function of c_1 and c_2 as reflected in Eq. (7).

$$v_i(t+1) = K(v_i(t) + R_1c_1(P_i - x_i(t)) + R_2c_2(P_g - x_i(t))), \tag{6}$$

where K is called constriction factor, given by

$$K = \frac{2}{\left|2 - \varphi + \sqrt{\varphi^2 - 4\varphi}\right|} \quad \text{where } \varphi = c_1 + c_2,\ \varphi > 4. \tag{7}$$

3. MCPSO algorithm

The initial inspiration for the PSO was the coordinated movement of swarms of animals in nature, e.g. schools of fish or flocks of birds. It reflects the cooperative relationship among the individuals within a swarm. However, in natural ecosystems, many species have developed cooperative interactions with other species to improve their survival. Such cooperative co-evolution is called symbiosis, firstly coined by German mycologist, Anton de Bary in 1879 [18]. The phenomenon of symbiosis can be found in all forms of life, from simple cells (e.g., eukaryotic organisms resulted probably from the mutualistic interaction between prokaryotes and some cells they infected) through to birds and mammals (e.g., African tick birds obtain a steady food supply by cleaning parasites from the skin of giraffes) [19].

According to the different symbiotic interrelationships, symbiosis can be classified into three main categories: mutualism (both species benefit by the relationship), commensalism (one species benefits while the other species is not affected), and parasitism (one species benefits and the other is harmed) [20]. We found that the commensalism model is suitable to be incorporated in the SPSO. Inspired by this research, a master–slave mode is incorporated into the SPSO, and the multi-swarm (species) cooperative optimizer (MCPSO) is thus developed.

In our approach, a population consists of one master swarm and several slave swarms. The symbiotic relationship between the master swarm and slave swarms can keep a right balance of exploration and exploitation, which is essential for the success of a given optimization task. The master–slave communication model, as shown in Fig. 1, is used to assign fitness evaluations and maintain algorithm synchronization.

In Fig. 1 each slave swarm executes a single PSO or its variants, including the update of position and velocity, and the creation of a new local population. When all the slave swarms are ready with the new generations, each slave swarm then sends the best local individual to the master swarm. The master swarm selects the best of all received individuals and evolves according to the following equations:

$$v_i^M(t+1) = wv_i^M(t) + R_1c_1(p_i^M - x_i^M(t)) + \Phi R_2c_2(p_g^M - x_i^M(t)) + (1-\Phi)R_3c_3(P_g^s - x_i^M(t)), \tag{8}$$

$$x_i^M(t+1) = x_i^M(t) + v_i^M(t). \tag{9}$$

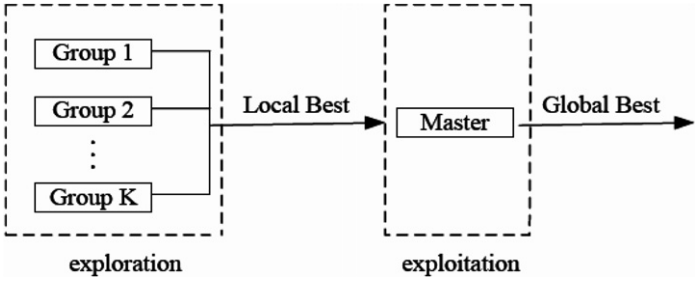

Fig. 1. The master–slave model.

Table 1
List of variables used in MCPSO

M	Master swarm
S	Slave swarm
R_3	Random number between 0 and 1
c_3	Acceleration constant
p_g^M	The best previous particle in master swarm
P_g^S	The best previous particle in slave swarms
$Gbest^M$	Fitness values determined by p_g^M
$Gbest^S$	Fitness values determined by P_g^S

For a minimization problem, Φ is a migration factor, given by

$$\Phi = \begin{cases} 0 & Gbest^S < Gbest^M, \\ 0.5 & Gbest^S = Gbest^M, \\ 1 & Gbest^S > Gbest^M. \end{cases} \tag{10}$$

The variables in Eqs. (7) and (9) are summarized in Table 1.

In the aforementioned version of MCPSO, each particle of the master swarm evolves by taking the combination of its own historical best information and that of all the swarms' (including the slave swarms) into account. In this way the master swarm enhances its particles based on direct competition with the slave swarms, i.e., the most fitted particle in all the swarms (master swarm or slave swarms) possesses the opportunity to guide the fight direction of the particles in the master swarm. Such a version is called as competitive version MCPSO, or COM-MCPSO.

Aside from this competitive version, a collaborative version MCPSO, called COL-MCPSO, is also proposed, in which the master swarm updates its particles, depending on its ability to collaborate with the slave swarms.

During flight, each particle of the master swarm adjusts its trajectory according to its own experience, the experience of its neighbors, and the experience of particles in the slave swarms, making use of the best previous position encountered by itself, its neighbors and the particles in the slave swarms. This idea is realized by further introducing a new term into the update equation of velocity component in SPSO. The resulting equations for the manipulation of the master swarm are

$$v_{id}^M = wv_{id}^M + R_1c_1(p_{id}^M - x_{id}^M) + R_2c_2(p_{gd}^M - x_{id}^M) + R_3c_3(p_{gd}^S - x_{id}^M), \tag{11}$$

$$x_{id}^M = x_{id}^M + v_{id}^M. \tag{12}$$

All the parameters adopted here are the same as those described in COM-MCPSO. The pseudo code of the MCPSO (COM-MCPSO and COL-MCPSO) algorithm is listed in Table 2.

Table 2
Pseudo code of the MCPSO algorithm

```
Algorithm MCPSO
Begin
  Initialize the population (population of the master and slave swarms)
  Evaluate the fitness value of each particle in the population
  Repeat
    Do in parallel
      Swarm i, 1 ⩽ i ⩽ S //S is the number of slave swarms
    End Do in parallel
    Barrier synchronization //wait for all processes to finish
    Select the fittest global individual (p_g^S or p_g^M) from all the swarms
    Evolve the mast swarm //Update the velocity and position using Eqs. (8) and (9) (or (11) and (12)), respectively
    Evaluate the fitness value of each particle
  Until a terminate-condition is met
End
```

4. Experimental studies

4.1. Benchmarks

A set of well-known benchmarks, that are commonly used in literature [21–25], were used to evaluate the performance, both in terms of solution quality and convergence rate, of the proposed algorithms. The benchmark problems used are a set of six non-linear functions, as minimization problems, which present different difficulties to the algorithms to be evaluated. These benchmark functions are

1. Sphere function

$$f_1(x) = \sum_i^n x_i^2. \tag{13}$$

2. Schwefel's problem 2.22

$$f_2(x) = \sum_i^n |x_i| + \prod_i^n |x_i|. \tag{14}$$

3. Rosenbrock function

$$f_3(x) = \sum_{i=1}^n 100 \times (x_{i+1} - x_i^2)^2 + (1 - x_i)^2. \tag{15}$$

4. Quartic function i.e. Noise

$$f_4 = \sum_{i=1}^n i x_i^4 + \text{rand}[0, 1). \tag{16}$$

5. Rastrigrin function

$$f_5(x) = \sum_{i=1}^n (x_i^2 - 10\cos(2\pi x_i) + 10). \tag{17}$$

6. Griewank function

$$f_6(x) = \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1. \tag{18}$$

Table 3
Parameters of the benchmark functions

Function	n	Minimum value	Range of search	Range of initialization
f_1	30	0	$[-100,100]^n$	$[50,100]^n$
f_2	30	0	$[-10,10]^n$	$[5,10]^n$
f_3	30	0	$[-100,100]^n$	$[15,30]^n$
f_4	30	0	$[-1.28,1.28]^n$	$[0.64,1.28]^n$
f_5	30	0	$[-5.12,5.12]^n$	$[2.56,5.12]^n$
f_6	30	0	$[-600,600]^n$	$[300,600]^n$

Table 3 lists the dimension, n, of each function, the ranges of their search space, and their global minimum fitnesses. It should be noted that a symmetric initialization method introduced by Angeline [11] is used here, in which the population is initialized only in a portion of the search space. This is to prevent a centre-seeking optimizer from "accidentally" finding the global optimum.

4.2. Experimental setting

The performance of two new methods (COM-MCPSO and COL-MCPSO) was compared with standard PSO (SPSO), as well as two variants of SPSO (RPSO and CPSO).

The parameters used for SPSO, RPSO CPSO were recommended in [3,16,23]. The maximum velocity $V_{\max}$ and minimum velocity $V_{\min}$ for SPSO, RPSO and CPSO were set at half value of the upper bound and lower bound, respectively.

The acceleration constants c_1 and c_2 for SPSO and CPSO were both 2.0. For RPSO, $c_1 = c_2 = 1.494$ was used. For MCSPO methods, a setting of $c_1 = c_2 = 2.05$, $c_3 = 2.0$ was adopted.

The inertia weight is critical for the convergence behavior of SPSO. A suitable value for the inertia weight usually provides a balance between global and local exploration abilities and consequently results in a better optimum solution. In our case, a decaying inertia weight starting at 0.9 and ending at 0.4 following Shi and Eberhart [4] was used for the SPSO and MCPSO methods. For CPSO, the constriction factor $K = 0.729$ was adopted.

The number of slave swarms S was set as 3 and three variants of SPSO were selected as evolution strategies for the slave swarms: global version of standard PSO (GSPSO), random inertial weight version of PSO (RPSO), constriction factor version of PSO (CPSO). All the parameters used in the slave swarms were the same as those defined above. For fair comparison, in all cases, the population size of SPSO and MCPSO was set at 80 (all the swarms of MCPSO include the same particles) and a fixed number of maximum generations 1000 were applied to all algorithms. A total of 50 runs for each experimental setting were conducted.

5. Experimental results

The experimental results (i.e. the best, worst, mean and standard deviation of the function values found in 50 runs) for each algorithm on each test function are listed in Table 4. In the table, the numbers in bold-face type represent the comparatively best values and all the results were reported as '0.000000e+00'. Moreover, the graphs presented in Figs. 2–7 illustrate the evolution of best fitness found by five algorithms, averaged for 50 funs for functions f_1–f_6.

With regards to the simpler function (relative to others) f_1 and f_2, all algorithms converged exponentially fast toward the fitness optimum. Since those two problems are unimodal functions, having only a single global minimum, fast convergence to the optimum is not a problem. However, only both MCPSO methods had particularly fast convergence, as can be seen from Fig. 3. This implies that the greater diversity provided by the slave swarms helps the MCPSO solve the problems quickly.

Function f_3 is a classic optimization problem, also known as banana function. The global optimum is inside a long, narrow, parabolic shaped flat valley. To find the valley is trivial, however convergence to the global optimum is difficult. So it is not surprisingly, SPSO converged slowly and ultimately produced poor average

Table 4
Results for all algorithms on benchmark functions

		COM-MPSO	COL-MCPSO	SPSO	CPSO	RPSO
f_1	Best	**1.4289e−036**	5.9114e−035	7.4172e−021	1.6297e−024	3.2041e−021
	Worst	**5.9499e−031**	3.2327e−030	2.5608e−015	2.5154e−017	6.8572e−012
	Mean	**3.8877e−032**	1.9008e−031	1.7536e−016	6.0619e−019	2.3795e−013
	Std	**1.0041e−031**	7.6249e−031	5.3120e−016	3.5620e−018	1.0571e−012
f_2	Best	**9.8761e−021**	1.2040e−019	2.7384e−012	1.6046e−014	1.9182e−011
	Worst	**9.4993e−018**	4.0084e−017	2.0818e−009	1.1565e−010	1.3310e−007
	Mean	**1.1696e−018**	5.2230e−018	1.4539e−010	9.8828e−012	1.3559e−008
	Std	**2.1867e−018**	1.2364e−017	3.5187e−010	2.1782e−011	2.7056e−008
f_3	Best	**6.6872e−005**	5.1000e−003	1.0560e−001	6.1000e−002	4.3780e−001
	Worst	2.2181e+001	**8.9301e+000**	8.5307e+002	1.5974e+002	2.5381e+002
	Mean	2.8349e+000	**2.5875e+000**	5.8515e+001	2.0250e+001	3.2367e+001
	Std	3.5827e+000	**2.0618e+000**	1.4576+002	3.9825+001	5.7988e+001
f_4	Best	**1.7097e−004**	2.8086e−004	1.4000e−003	4.1943e−004	3.4789e−004
	Worst	**3.8000e−003**	6.3000e−003	2.0600e−002	8.8000e−003	9.1000e−003
	Mean	**1.5000e−003**	1.7000e−003	7.0000e−003	2.8000e−003	3.0000e−003
	Std	**9.3850e−004**	1.0000e−003	4.6000e−003	2.0000e−003	2.1000e−003
f_5	Best	**8.5123e−012**	2.4705e−009	7.8337e−010	2.9849e+000	1.9899e+000
	Worst	**1.9899e−000**	3.9798e+000	6.9647e+000	4.0793e+001	1.9900e+001
	Mean	**6.9850e−001**	1.5561e+000	3.7228e+000	1.4632e+001	8.5567e+000
	Std	**7.2840e−001**	1.0855e+000	1.5131e+000	1.0297e+001	5.6043e+000
f_6	Best	**1.4800e−002**	2.2100e−002	2.9600e−002	2.7000e−002	2.4600e−002
	Worst	**7.8700e−002**	8.8500e−002	2.6810e−001	2.0680e−001	1.5520e−001
	Mean	**4.9100e−002**	5.6500e−002	8.3600e−002	9.0800e−002	8.7900e−002
	Std	**1.7800e−002**	1.9500e−002	4.6100e−002	3.7100e−002	3.7200e−002

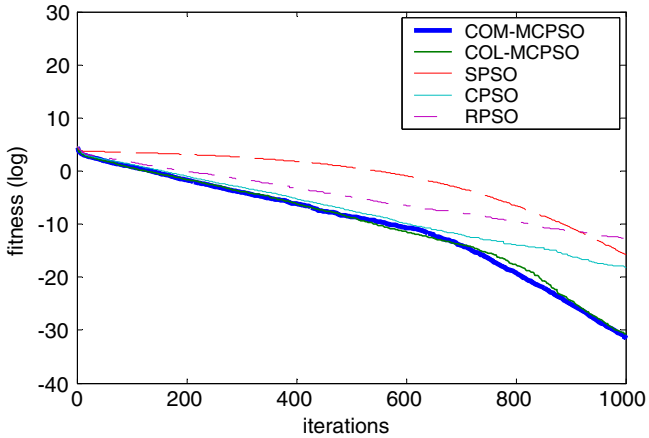

Fig. 2. f_1, Sphere function.

best fitness for this case. In contrast, RPSO and CPSO method improved the average optimum solution significantly when compared with SPSO, but they still performed worse than MCPSO methods. Further, with the MCPSO concept, the standard deviation of the final solution for 50 trials was found to be significantly low for function f_3 compared with all of other methods, as shown in Table 3.

Function f_4 is a noisy function. All algorithms seem to converge in a similar pattern, see Fig. 5. The COM-MCPSO had the best convergence speed, followed by COL-MCPSO, CPSO, RPSO and SPSO.

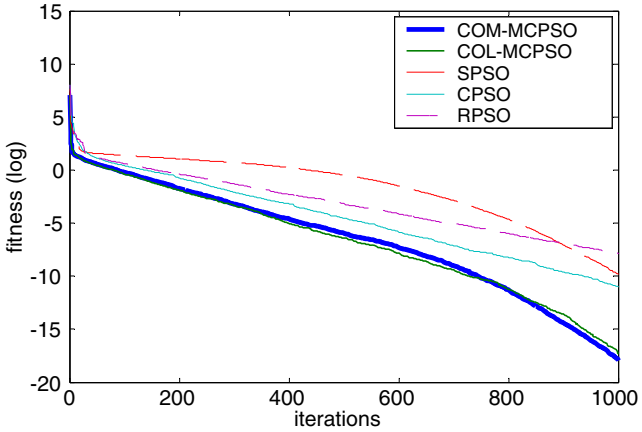

Fig. 3. f_2, Schwefel's problem 2.22.

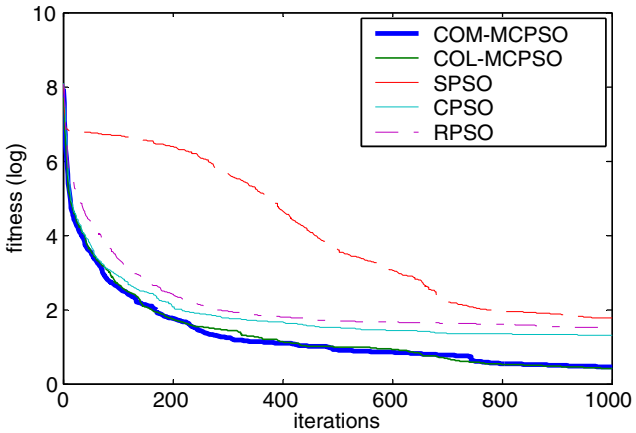

Fig. 4. f_3, Generalized Rosenbrock function.

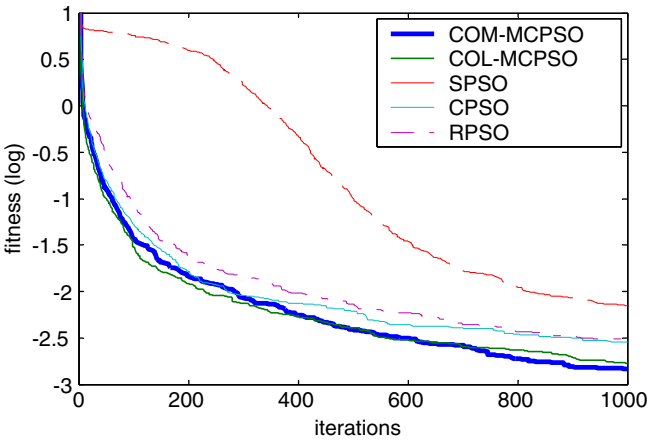

Fig. 5. f_4, Quartic function i.e. noise.

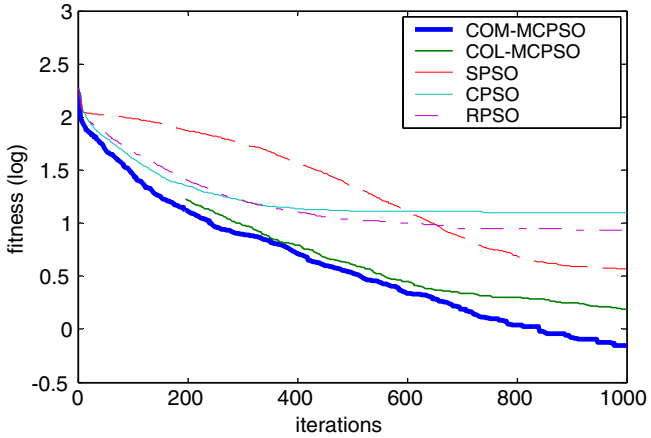

Fig. 6. f_5, Generalized Rastrigrin's function.

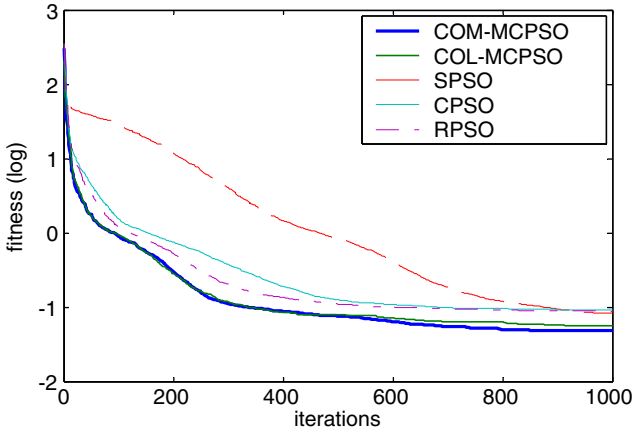

Fig. 7. f_6, Generilized Griewank function.

Function f_5 is a highly multi-modal function when in 10 dimensions or less. Regarding this case there was an interesting tendency. The search improvements of the SPSO were a bit slower but eventually better compared to RPSO and CPSO. MCPSO method clearly performed best and gave consistently a near-optimum result, whereas other PSO methods stagnated and flatted out with no further improvement.

Function f_6 exhibited a pattern similar to that observed with function f_5. Both RPSO and CPSO converged very fast to good values near the optimum, but struggled with premature convergence after 600 generations. The SPSO converged slowly, but outperformed CPSO and RPSO after 1000 generations. Again, MCPSO method achieved the fastest search improvements and yielded promising end-results, while other algorithms failed many times to reach a solution as good.

In a general analysis of the values in Table 4 and the graphs in Figs. 2–7 the following major conclusions can be drawn:

(i) COM-MCPSO is highly competitive with COL-MCPSO on all of the problems, usually surpassing its performance. The differences between the results generated by COM-MCPSO and COL-MCPSO are not statistically significant.
(ii) COM-MCPSO has shown to have improved the solution consistently throughout the simulation studies and has found better solution for most of the functions. The only exception is about function f_2, for which COL-MCPSO seems to fine-tune its results slightly better than COM-MCPSO.

(iii) On the multi-modal functions (f_5 and f_6), RPSO converges significantly faster than SPSO at the early stage of the optimization process, but it was outperformed by SPSO at the latter stages of the process.
(iv) CPSO performed better than SPSO on all the unimodal benchmarks (f_1–f_3), while SPSO gives better performance on the multi-modal benchmarks (f_5 and f_6).
(v) In all the test cases, both CPSO and RPSO seem to converge in a similar pattern, see Figs. 2–7.

From this summary, it can be concluded that MCPSO is clearly the best performing algorithm in this study. Therefore, the MCPSO method can be identified as a highly consistent strategy in finding the optimum solution compared with other methods. We believe that the better performance observed in those functions for MCPSO is the result of diversity maintained in the multi-swarm evolutionary process. But this indicator is not enough – we need to assess also the robustness of the algorithm, which may be evaluated by the stability of the results. Figs. 8–13 show the multi-swarm co-evolutionary process for all of the functions by COM-MCPSO and COL-MCPSO, respectively. It should be mentioned that we present those graphs only from a single test run.

In each case, we found that the slave swarms performed well in initial iterations but failed to make further progress in later iterations. The experiments with the Rosenbrock function illustrate this point clearly. There was a quick decrease in fitness, in initial iterations, for the master swarm and the slave swarms probably when the particles where descending the walls of the function valley. When the plateau was reached, the slave swarms started to produce improvements very slowly as usually happens with this function while the master swarm kept finding better solutions long after the stagnation of the slave swarms. By looking at the shapes of

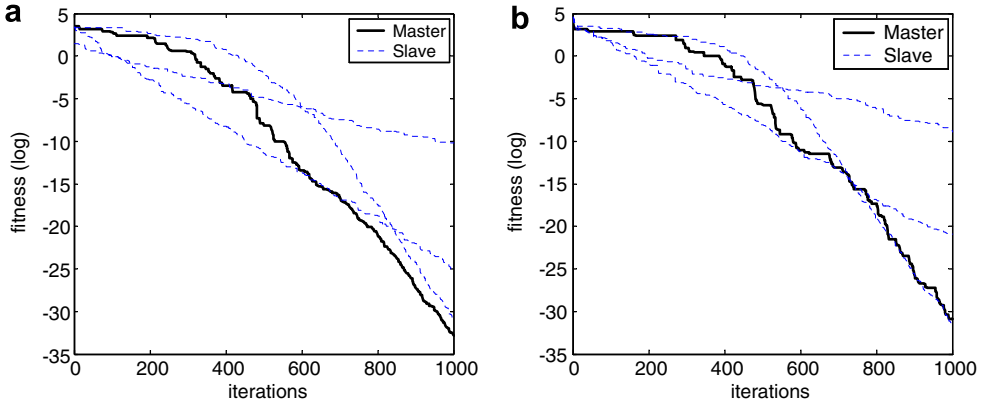

Fig. 8. Co-evolutionary process for f_1 using: (a) COM-MCPSO; (b) COL-MCPSO.

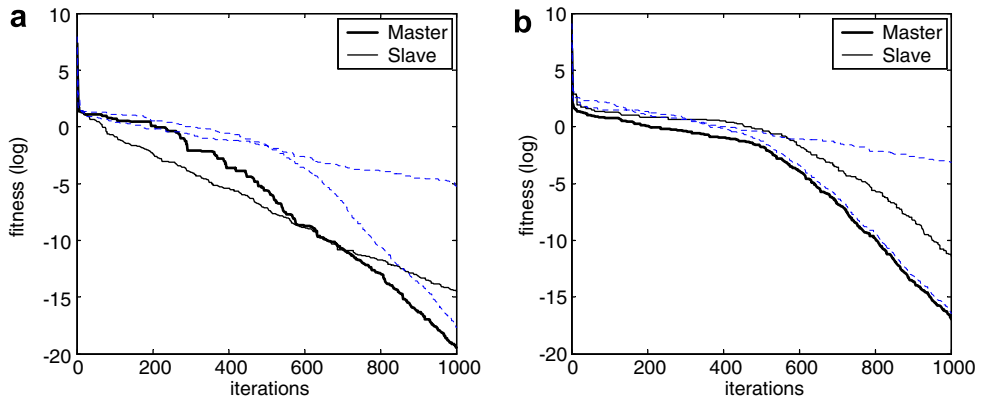

Fig. 9. Co-evolutionary process for f_2 using: (a) COM-MCPSO; (b) COL-MCPSO.

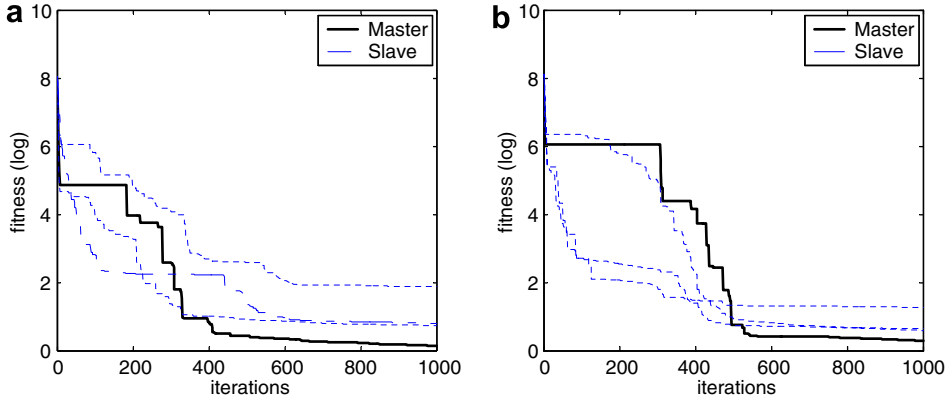

Fig. 10. Co-evolutionary process for f_3 using: (a) COM-MCPSO; (b) COL-MCPSO.

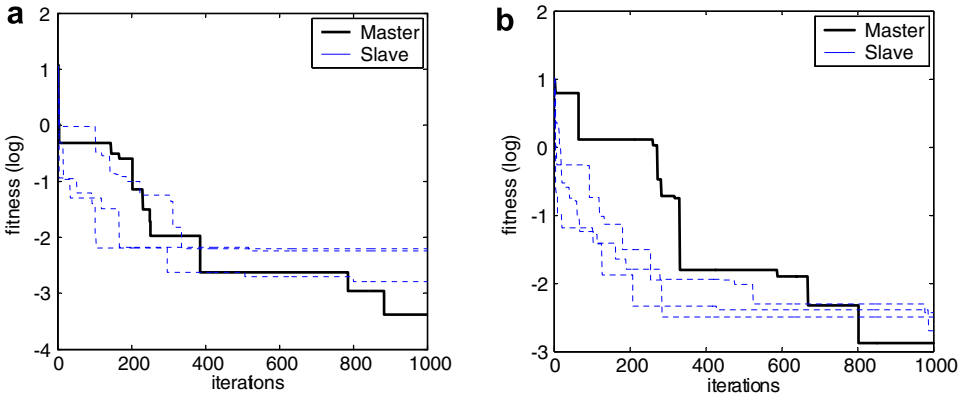

Fig. 11. Co-evolutionary process for f_4 using: (a) COM-MCPSO; (b) COL-MCPSO.

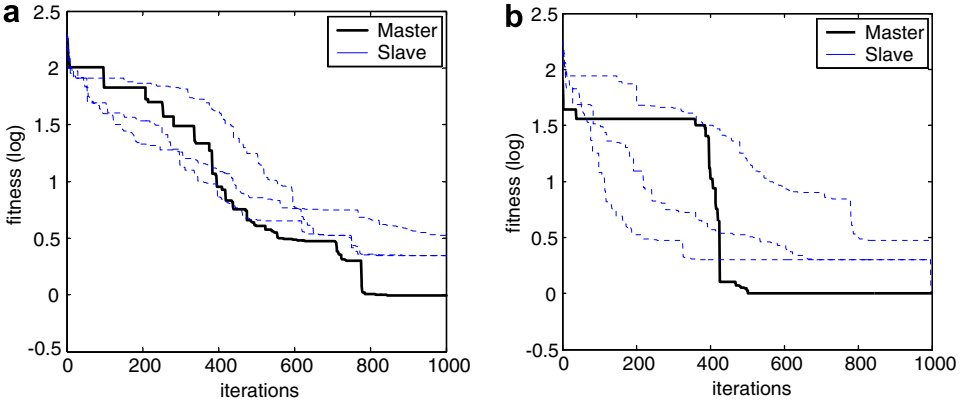

Fig. 12. Co-evolutionary process for f_5 using: (a) COM-MCPSO; (b) COL-MCPSO.

the curves in the graphs, it is easy to see that each particle in the master swarm can keep track of the previous best position found by the slave swarms, as well as find a better position based on its own knowledge. In fact, since the competition relationship of the slave swarms, the master swarm would not be influenced much when

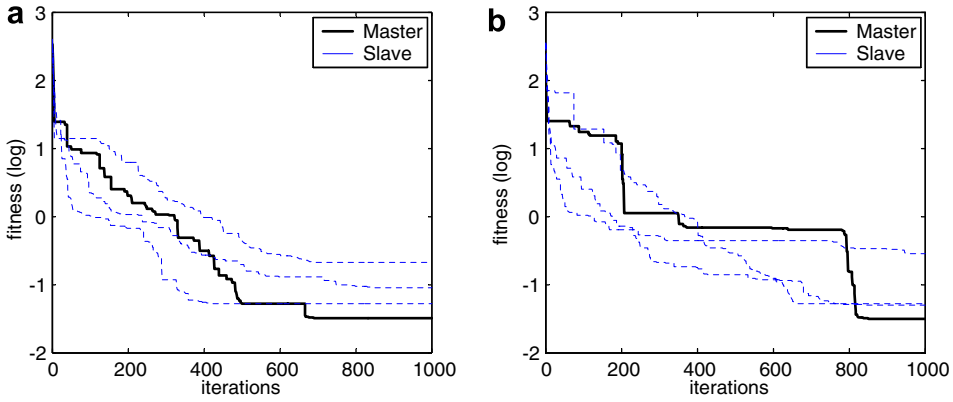

Fig. 13. Co-evolutionary process for f_6 using: (a) COM-MCPSO; (b) COL-MCPSO.

a certain slave swarms got stuck at a local optima. It may be concluded that the results generated by MCPSO is robust.

6. Conclusions and future work

In this paper, a multi-swarm cooperative particle swarm optimizer has been proposed to improve the performance of SPSO. MCPSO is a master–slave model that consists of one master swarm and several slave swarms. The evolution of slave swarms is likely to amplify the diversity of individuals of the population and consequently to generate more promising particles for the master swarm. The master swarm updates the particle states based on both its own experience and that of the most successful particles in the slave swarms.

Based on the co-evolutionary relationship between master swarm and slave swarms, two versions of MCPSO have been studied: the competitive version of MCPSO (COM-MCPSO) and the collaborative version of MCPSO (COL-MCPSO). Compared with SPSO and its variants on six benchmark functions, the proposed method is less susceptible to premature convergence and less likely to be stuck in local optima. The preliminary results suggest that MCPSO have superior features, both in high quality of the solution and robustness of the results.

Future work will focus on optimizing the performance of the MCPSO. In addition, extensive study of the applications in more complex practical optimization problems is necessary to fully investigate the properties and evaluate the performance of MCPSO.

Acknowledgements

The work is supported by the National Natural Science Foundation of China under contract No.70431003 and the National Basic Research Program of China under contract No. 2002CB312200.

References

[1] R.C. Eberchart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of the 6th International Symposium on Micromachine and Human Science, Nagoya, Japan, 1995, pp. 39–43.
[2] J. Kennedy, R.C. Eberchart, Particle swarm optimization, in: Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, 1995, pp. 1942–1948.
[3] Y. Shi, R.C. Eberhart, Empirical study of particle swarm optimization, in: Proceedings of Congress on Evolutionary Computation, Washington, DC, 1999, pp. 1945–1949.
[4] Y. Shi, R.C. Eberhart, A modified particle swarm optimizer, in: Proceedings of IEEE International Conference on Evolutionary Computation, Anchorage, AK, May 1998, pp. 69–73.
[5] J. Kennedy, R.C. Eberchart, Y. Shi, Swarm Intelligence, Morgan Kaufmann Publishers, San Francisco, 2001.

[6] R. Mendes, P. Cortez, M. Rocha, J. Neves, Particle swarms for feedforward neural network training, in: Proceedings of the International Joint Conference on Neural Networks (IJCNN 2002), 2002, pp. 1895–1899.
[7] G.K. Venayagamoorthy, S. Doctor, Navigation of mobile sensors using PSO and embedded PSO in a fuzzy logic controller, in: Proceedings of the 39th IEEE IAS Annual Meeting on Industry Applications, Seattle, USA, 2004, pp. 1200–1206.
[8] K.E. Parsopoulos, E.I. Papageorgiou, P.P. Groumpos, M.N Vrahatis, A first study of fuzzy cognitive maps learning using particle swarm optimization, in: Proceedings of IEEE Congress on Evolutionary Computation 2003 (CEC 2003), Canbella, Australia, 2003, pp. 1440–1447.
[9] M.A. Abido, Optimal power flow using particle swarm optimization, Int. J. Elect. Power Energy Syst. 24 (7) (2002) 563–571.
[10] V. Tandon, H. El-Mounayri, H. Kishawy, NC end milling optimization using evolutionary computation, Int. J. Mach. Tools Manuf. 42 (2002) 595–605.
[11] P.J. Angeline, Evolutionary optimization versus particle swarm optimization and philosophy and performance difference, in: Proceedings of 7th Annual Conference on Evolutionary Programming, San Diego, USA, 1998, pp. 601–610.
[12] X.H. Shi, Y.C. Liang, H.P. Lee, C. Lu, L.M. Wang, An improved GA and a novel PSO-GA-based hybrid algorithm, Inform. Process. Lett. 93 (2005) 255–261.
[13] W.J. Zhang, X.F. Xie, DEPSO: hybrid particle swarm with differential evolution operator, in: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Washington, DC, USA, 2003, pp. 3816–3821.
[14] S. He, Q.H. Wu, J.Y. Wen, J.R. Saunders, R.C. Paton, A particle swarm optimizer with passive congregation, Biosystems 78 (2004) 135–147.
[15] B. Niu, Y.L. Zhu, X.X. He, Multi-population cooperative particle swarm optimization, Lect. Notes Art. Intell. 3630 (2005) 874–883.
[16] R.C. Eberhart, Y. Shi, Tracking and optimizing dynamic systems with particle swarms, in: Proceedings of IEEE Congress on Evolutionary Computation, Seoul, Korea, 2001, pp. 94–97.
[17] M. Clerc, J. Kennedy, The particle swarm: explosion, stability, and convergence in a multidimensional complex space, IEEE Trans. Evolut. Comput. 6 (2002) 58–73.
[18] V. Ahmadjian, S. Paracer, Symbiosis: An Introduction to Biological Associations, Oxford University Press, New York, 2000.
[19] J. Sapp, The dynamics of symbiosis: an historical overview, Canadian J. Bot. 82 (2004) 1–11.
[20] A.E. Douglas, Symbiotic Interactions, Oxford University Press, Oxford, 1994.
[21] J. Vesterstrøm, R. Thomsen, A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems, in: Proceedings of the 2004 Congress on Evolutionary Computation, Portland, Oregon, USA, 2004, pp. 1980–1987.
[22] P.J. Angeline, Using selection to improve particle swarm optimization, in: Proceedings of IEEE International Conference on Computational Intelligence, 1998, pp. 84–89.
[23] R.C. Eberhart, Y. Shi, Comparing inertia weights and constriction factors in particle swarm optimization, in: Proceedings of IEEE International Congress on Evolutionary Computation, San Diego, CA, 2000, pp. 84–88.
[24] Y. Shi, R.C. Eberhart, Fuzzy adaptive particle swarm optimization, in: Proceedings of IEEE International Congress on Evolutionary Computation, Seoul, Korea, 2001, pp. 101–106.
[25] P.N. Suganthan, Particle swarm optimizer with neighborhood operator, in: Proceedings of IEEE International Congress on Evolutionary Computation, Washington, DC, USA, 1999, pp. 1958–1962.