Front. Comput. Sci. China 2009, 3(1): 38–52
DOI 10.1007/s11704-009-0010-x

RESEARCH ARTICLE

A hybrid multi-swarm particle swarm optimization to solve constrained optimization problems

Yong WANG (✉), Zixing CAI

School of Information Science and Engineering, Central South University, Changsha 410083, China

Abstract In the real-world applications, most optimization problems are subject to different types of constraints. These problems are known as constrained optimization problems (COPs). Solving COPs is a very important area in the optimization field. In this paper, a hybrid multi-swarm particle swarm optimization (HMPSO) is proposed to deal with COPs. This method adopts a parallel search operator in which the current swarm is partitioned into several sub-swarms and particle swarm optimization (PSO) is severed as the search engine for each sub-swarm. Moreover, in order to explore more promising regions of the search space, differential evolution (DE) is incorporated to improve the personal best of each particle. First, the method is tested on 13 benchmark test functions and compared with three state-of-the-art approaches. The simulation results indicate that the proposed HMPSO is highly competitive in solving the 13 benchmark test functions. Afterward, the effectiveness of some mechanisms proposed in this paper and the effect of the parameter setting were validated by various experiments. Finally, HMPSO is further applied to solve 24 benchmark test functions collected in the 2006 IEEE Congress on Evolutionary Computation (CEC2006) and the experimental results indicate that HMPSO is able to deal with 22 test functions.

Keywords constrained optimization problems, constraint-handling technique, particle swarm optimization, differential evolution

Received September 1, 2008; accepted October 29, 2008

E-mail: {ywang, zxcai}@mail.csu.edu.cn

1 Introduction

We are interested in solving the general *constrained optimization problems* (COPs) formulated as follows (in the minimization sense):

$$\begin{aligned} \text{minimize} \quad & f(\vec{x}), \vec{x} = (x_1, x_2, \ldots, x_n) \in \mathcal{R}^n \\ \text{subject to} \quad & g_j(\vec{x}) \leqslant 0, j = 1, 2, \ldots, q \\ & h_j(\vec{x}) = 0, j = q+1, q+2, \ldots, m \end{aligned}$$

where $\vec{x} \in \Omega \subseteq S$, Ω is the feasible region defined by a set of m additional linear and nonlinear constraints: $\Omega = \{\vec{x} \mid g_j(\vec{x}) \leqslant 0, j = 1, 2, \ldots, q; h_j(\vec{x}) = 0, j = q+1, q+2, \ldots, m; \vec{x} \in S\}$, S is the decision space defined by the parametric constraints:$L_i \leqslant x_i \leqslant U_i, 1 \leqslant i \leqslant n$, $g_j(\vec{x})$ is the jth inequality constraint, and $h_j(\vec{x})$ is the jth equality constraint.

If an inequality constraint satisfies $g_j(\vec{x}) = 0 (j \in \{1, 2, \ldots, q\})$ at any point $\vec{x} \in \Omega$, we say it is *active* at $\vec{x}$. All equality constraints $h_j(\vec{x}) (j = q+1, q+2, \ldots, m)$ are considered *active* at all points of Ω.

Usually, the degree of constraint violation of individual $\vec{x}$ on the jth constraint is calculated as follows:

$$G_j(\vec{x}) = \begin{cases} \max\{0, g_j(\vec{x})\}, & 1 \leqslant j \leqslant q, \\ \max\{0, |h_j(\vec{x})| - \varepsilon\}, & q+1 \leqslant j \leqslant m, \end{cases} \quad (1)$$

where ε is a positive tolerance value for equality constraints. Then, $G(\vec{x}) = \sum_{j=1}^{m} G_j(\vec{x})$ reflects the degree of constraint violation of the individual $\vec{x}$.

During the past decade, solving COPs based on evolutionary algorithms (EAs) has attracted much attention [1, 2]. When using EAs to solve COPs, the way that the con-

straints are handled is very important, since EAs are essentially unconstrained search techniques and lack an explicit mechanism to bias the search in constrained search spaces [3]. Currently, the most popular constraint-handling techniques include: methods based on penalty functions, methods based on preference of feasible solutions over infeasible solutions, and methods based on multi-objective optimization techniques. Methods based on penalty functions are the most common techniques to deal with constraints because of the ease to implement, the principal idea of which is to add penalty term into the objective function and, therefore, to convert COPs into unconstrained optimization problems. With respect to methods based on preference of feasible solutions over infeasible solutions, feasible solutions are always considered better than infeasible solutions. The main idea of methods based on multi-objective optimization techniques is that after converting COPs into unconstrained multi-objective optimization problems, multi-objective optimization techniques are exploited to tackle the converted problems [4−7].

Particle swarm optimization (PSO) is a new branch in the area of evolutionary computation introduced by Kennedy and Eberhart in 1995 [8]. PSO is easy to implement and has been empirically verified to perform well on many optimization problems. Because of the success of PSO in solving unconstrained optimization problems, PSO has gradually gained attention in sloving COPs in the last few years. Liang and Suganthan [9] proposed a dynamic multi-swarm PSO with a novel constraint-handling mechanism for solving COPs. In their proposal, the population is periodically and randomly divided into several sub-swarms. In addition, the objective and constraints are assigned to the sub-swarms adaptively according to the difficulties of the constraints. As a result, the constraints that are more difficult will have more sub-swarms working for it, and the constraints that are easier will have less sub-swarms working for it. Moreover, sequential quadratic programming (SQP) is used for local search. Krohling and Coelho [10] presented a co-evolutionary PSO based on Gaussian distribution for constrained optimization, in which the accelerating coefficients of PSO are generated using a Gaussian probability distribution. Furthermore, two populations are evolved in this method: one for the variable vector and the other for the lagrange multiplier vector. He and Wang [11] also proposed a co-evolutionary PSO. In this approach, the notion of co-evolution is employed to cope with both decision variables and constraints. Two swarms evolve interactively using PSO, one is used for searching good solutions and the other is used for optimizing appropriate penalty factors. Recently, He and Wang [12] introduced a hybrid PSO with a feasibility-based rule [13] to solve COPs. In this hybrid method, the feasibility-based rule is employed to update the personal best of each particle. Since the feasibility-based rule always considers feasible solutions better than infeasible solutions, simulated annealing is applied to the best solution of the swarm to avoid premature convergence. Pulido and Coello Coello [14] proposed a simple mechanism to handle constraints with PSO. In this mechanism, if both particles compared are infeasible, the particle with the lower value in its normalized violation of constraints wins. Besides, a turbulence operator is incorporated to improve the exploratory capability of PSO. Another novel method called PESO+ (Particle Evolutionary Swarm Optimization Plus) is proposed by Munoz-Zavala et al. [15]. The PESO+ includes two main features: 1) two perturbation operators, the aim of which is to alter the memory of best location of the particles; and 2) an external file which is intended to preserve feasible solutions at different tolerance values of equality constraints. Zielinski and Laur [16] combined PSO with a simple constraint-handling technique to solve COPs.

Recently, Wang et al. [6] proposed a multi-objective optimization based hybrid evolutionary algorithm to solve COPs. This method consists of two main parts: the global search model and the local search model. In the global search model, a niching genetic algorithm is proposed. The local search model performs a parallel local search operator based on the clustering partition of the population. The comparison of individuals is based on multi-objective optimization technique in this method. Inspired by Ref. [6], we propose a hybrid multi-swarm particle swarm optimization (HMPSO) for constrained optimization in this design. At each generation, the swarm is split into several sub-swarms and each sub-swarm evolves independently by taking advantage of PSO as the search algorithm. In addition, the personal best of each particle are updated by differential evolution (DE) so as to further enhance the global search ability of PSO. Unlike Ref. [6], HMPSO uses the feasibility-based rule to compare particles in the swarm. HMPSO is exploited to solve 13 benchmark test functions, and the empirical results suggest that the approach proposed is capable of yielding competitive results with respect to three state-of-the-art methods in constrained evolutionary optimization. Afterward, the effectiveness and efficiency of HMPSO have been further validated by 24 benchmark test functions in CEC2006.

The remainder of this paper is organized as follows. Section 2 introduces the basic ideas of PSO and DE. Section 3 is dedicated to explain the HMPSO in detail. The experimental results are presented and compared in Section 4. Section 5 discusses the effectiveness of some mechanisms and the ef-

fect of the parameter setting. Moreover, the performance of HMPSO is also validated on 24 benchmark test functions in CEC2006. Finally, Section 6 concludes this paper.

2 The basic ideas of PSO and DE

2.1 The basic idea of PSO

As a stochastic global optimization method, PSO takes inspiration of the motion of a flock of birds. In PSO, each potential solution is regarded as a particle. A set which is composed of particles is called a swarm. The movement of particles in the search space is dynamically influenced by their personal past experience and successful experience attained by their neighbors or the whole swarm. At the tth generation, the position vector and the velocity vector of the ith particle in the n-dimensional search space can be represented as $\vec{x}_i^t = (x_{i,1}^t, \ldots, x_{i,n}^t)$ and $\vec{v}_i^t = (v_{i,1}^t, \ldots, v_{i,n}^t)$. In addition, at the tth generation $pbest_i^t = (pbest_{i,1}^t, \ldots, pbest_{i,n}^t)$ is denoted as the best previous position of the ith particle, and $gbest^t = (gbest_1^t, \ldots, gbest_n^t)$ is denoted as the best position of the whole swarm. During the evolution, Eq. (2) updates each dimension of the velocity for the ith particle for the next generation, whereas Eq. (3) updates the ith particle's position in the search space:

$$v_{i,j}^{t+1} = v_{i,j}^t + cr_1(pbest_{i,j}^t - x_{i,j}^t) + cr_2(gbest_j^t - x_{i,j}^t), \quad (2)$$

$$x_{i,j}^{t+1} = x_{i,j}^t + v_{i,j}^{t+1}, \quad (3)$$

where $j \in \{1, 2, \ldots, n\}$, c is a constant known as acceleration coefficient, r_1 and r_2 are two separately generated uniformly distributed random numbers in the range [0,1].

It is necessary to note that in terms of the above version of PSO, since it lacks the mechanism to control the magnitude of the velocities, the velocities and position of the particles might careen toward infinity, which results in a kind of explosion and divergence [17]. To overcome this defect, the velocities are usually confined within the range of $[-\vec{V}_{\max}, \vec{V}_{\max}]$.

Shi and Eberhart [18] addressed the aforementioned explosion problem by further incorporating a weight parameter into the previous velocity of the particle. As a result, the velocity and position updates are:

$$v_{i,j}^{t+1} = wv_{i,j}^t + c_1r_1(pbest_{i,j}^t - x_{i,j}^t) + c_2r_2(gbest_j^t - x_{i,j}^t), \quad (4)$$

$$x_{i,j}^{t+1} = x_{i,j}^t + v_{i,j}^{t+1}, \quad (5)$$

where w denotes the inertia weight, c_1 and c_2 are the acceleration constants, r_1 and r_2 are two separately generated uniformly distributed random numbers in the range [0,1].

Clerc and Kennedy [19] proposed an alternative version of PSO in which the velocity adjustment is given as follows:

$$v_{i,j}^{t+1} = \chi(v_{i,j}^t + c_1r_1(pbest_{i,j}^t - x_{i,j}^t) + c_2r_2(gbest_j^t - x_{i,j}^t)), \quad (6)$$

where

$$\chi = \frac{2}{|2 - \phi - \sqrt{\phi^2 - 4\phi}|}, \quad (7)$$

and $\phi = c_1 + c_2$, $\phi > 4$. By making use of the constriction coefficient χ, the above version of PSO eliminates the parameter $\vec{V}_{\max}$.

By analyzing Eq. (6), Krohling and Coelho [10] concluded that the mean values of the two stochastic coefficients for the local term $(pbest_{i,j}^t - x_{i,j}^t)$ and the global term $(gbest_j^t - x_{i,j}^t)$ lies in the interval [0.72, 0.86]. Consequently, a proper choice for the probability distribution that generates stochastic coefficients is the absolute value of the Gaussian probability distribution with zero mean and unit variance, i.e., $abs(N(0,1))$ and, as a result, the velocity equation is updated according to:

$$v_{i,j}^{t+1} = |randn|(pbest_{i,j}^t - x_{i,j}^t) + |Randn|(gbest_j^t - x_{i,j}^t), \quad (8)$$

where $|randn|$ and $|Randn|$ are positive random numbers generated using $abs(N(0,1))$.

2.2 The basic idea of DE

Differential evolution (DE) was proposed by Storn and Price [20] in 1995 which is a stochastic and population-based optimization algorithm, and uses vector differences to perturb the vector population.

Initially, DE starts with a population consisting of N n-dimensional vectors which have the form:

$$\vec{x}_i^t = \{x_{i,1}^t, x_{i,2}^t, \ldots, x_{i,n}^t\}, \quad i = 1, 2, \ldots, N, \quad (9)$$

where t denotes the generation number. These vectors are randomly selected on the intervals: $[L_i, U_i]$, $i = 1, 2, \ldots, n$. During the evolution, each of the N vectors undergoes mutation, crossover and selection operations.

There are several versions for DE. Next, the most popular version of DE called "DE/rand/1/bin" will be introduced.

Mutation operation: For each given vector $\vec{x}_i^t$ at generation t, three vectors $\vec{x}_{r1}^t$, $\vec{x}_{r2}^t$, and $\vec{x}_{r3}^t$ are randomly selected such that the indices i, r_1, r_2 and r_3 are distinct. $\vec{x}_i^t$ is called the target vector. Afterward, the weighted difference of two of the vectors is added to the third to form a mutant vector $\vec{v}_i^t = \{v_{i,1}^t, v_{i,2}^t, \ldots, v_{i,n}^t\}$:

$$\vec{v}_i^t = \vec{x}_{r1}^t + F(\vec{x}_{r2}^t - \vec{x}_{r3}^t). \quad (10)$$

Crossover operation: The trial vector $\vec{u}_i^t$ is developed from the elements of the target vector, $\vec{x}_i^t$ and the elements of the mutant vector $\vec{v}_i^t$, using a "binomial" crossover operation:

$$u_{i,j}^t = \begin{cases} v_{i,j}^t, & \text{if } rand_j \leqslant C_r \text{ or } j = j_{rand}, \\ x_{i,j}^t, & \text{otherwise,} \end{cases} \tag{11}$$

where $i = 1, 2, \ldots, N$, $j = 1, 2, \ldots, n$, index j_{rand} is a randomly chosen integer within the range $[1, n]$, $rand_j$ is the jth evaluation of a uniform random number generator, and $C_r \in (0, 1)$. The condition "$j = j_{rand}$" ensures that the trial vector $\vec{u}_i^t$ get at least one element from its mutant vector $\vec{v}_i^t$.

Selection operation: The trial vector $\vec{u}_i^t$ is compared with the target vector $\vec{x}_i^t$ and the one with the lowest objective function value are survived into the next generation:

$$\vec{x}_i^{t+1} = \begin{cases} \vec{u}_i^t, & \text{if } f(\vec{u}_i^t) \leqslant f(\vec{x}_i^t), \\ \vec{x}_i^t, & \text{otherwise.} \end{cases} \tag{12}$$

It is important to note that in DE mutation, crossover and selection operations continue over the course of evolution until some stopping criterion is reached. In addition, there are three control parameters in DE, i.e., the population size N, the scaling factor F and the crossover constant C_r.

3 Hybrid multi-swarm particle swarm optimization (HMPSO)

There are two main versions in the PSO domain: global PSO and local PSO. In the local version of PSO, each particle's velocity is adjusted based on its personal best and the best performance achieved with its neighborhood rather than the entire swarm. Since the method proposed in this paper adopts multi-swarm to search the global optimum, the local version is used. In addition, because of the ease of implementation, less parameters and good performance, Krohling and Coelho's PSO [10] is applied as the search engine. Therefore, in this paper the velocity $v_{i,j}^t$ and position $x_{i,j}^t$ updates of the jth dimension of the ith particle become:

$$v_{i,j}^{t+1} = |randn|(pbest_{i,j}^t - x_{i,j}^t) + |Randn|(lbest_{i,j}^t - x_{i,j}^t), \tag{13}$$

$$x_{i,j}^{t+1} = x_{i,j}^t + v_{i,j}^{t+1}, \tag{14}$$

where $\overrightarrow{lbest}_i^t = (lbest_{i,1}^t, \ldots, lbest_{i,n}^t)$ is the best position achieved with its neighbors.

HMPSO works as follows. Initially, the swarm P_0 of size N is randomly and uniformly selected between the lower and upper bounds defined for each variable. At each generation, the swarm is first split into several sub-swarms and each swarm evolves in parallel. After that, the personal best of each particle is updated by DE. The general framework of HMPSO is depicted in Fig. 1. The two main procedures of HMPSO, i.e., swarm's splitting and sub-swarms' evolution and updating the personal best by DE, will be illustrated in the sequel.

3.1 Swarm's splitting and sub-swarms' evolution

Figure 2 summarizes the steps for partitioning swarm and evolving sub-swarms.

First, the particles in the swarm are listed according to the following criterion: 1) the feasible solutions are listed in front of the infeasible solutions; 2) the feasible solutions are sorted in ascending order of their objective function values; and 3) the infeasible solutions are sorted in ascending order of their degree of constraint violations.

Afterward, the first particle in the swarm is selected as the local best ($\overrightarrow{lbest}$) of the first sub-swarm. Since each member in the sub-swarm should directly learn the experience from the local best, if the members are too close to the local best, the effect of the learning might not be very significant. On the other hand, if all particles of the sub-swarm are located in a small region of the search space, the global search capabil-

Step 1 $t = 0$;
Step 2 randomly generate an initial swarm P_0 which consists of N particles, i.e., $\vec{x}_1^0, \ldots, \vec{x}_N^0$;
Step 3 calculate $f(\vec{x}_i^0)$ and $G(\vec{x}_i^0)$ for each particle, $i = 1, 2, \ldots, N$;
Step 4 record particles'personal bests, i.e., $\overrightarrow{pbest}_1^0, \ldots, \overrightarrow{pbest}_N^0$;
Step 5 **while** (termination condition=false) **do**
Step 6 divide the swarm into several sub-swarms, and each sub-swarm evolves independently using Eqs. (13) and (14);
Step 7 update the personal best of each particle using DE;
Step 8 $t = t + 1$;
Step 9 **end while**

Fig. 1 The general framework of HMPSO

Algorithm: swarm's splitting and sub-swarms' evolution

Input: $P = \{\vec{x}_1, \ldots, \vec{x}_N\}$, P' =Ø, $Pbest = \{\overrightarrow{pbest}_1, \ldots, \overrightarrow{pbest}_N\}$, $Pbest'$= Ø

Step 1 $k = 0$;

Step 2 sort the swarm according to the following criteria: 1) the feasible solutions are listed in front of the infeasible solutions; 2) the feasible solutions are sorted in ascending order of their objective function values; and 3) the infeasible solutions are sorted in ascending order of their degree of constraint violations.

Step 3 **while** $k < \lfloor N/N_s \rfloor$ **do**

Step 4 $\vec{y}_1$= the first particle in the swarm; /* under this condition, $\vec{y}_1$ is the local best of the sub-swarm */

Step 5 find out the $(N_s - 1)$ particles (denoted as $\vec{y}_2, \ldots, \vec{y}_{N_s}$) with the largest Euclidean distance from $\vec{y}_1$; /* $\vec{y}_2, \ldots, \vec{y}_{N_s}$ are the other members of the sub-swarm */

Step 6 let $M\_1 = \{\vec{y}_1, \ldots, \vec{y}_{N_s}\}$. Suppose that the personal bests of the set M_1 are denoted as $\overrightarrow{ybest}_1, \ldots, \overrightarrow{ybest}_{N_s}$, let $M\_2 = \{\overrightarrow{ybest}_1, \cdots, \overrightarrow{ybest}_{N_s}\}$;

Step 7 $P = P \backslash M\_1$ and $Pbest = Pbest \backslash M\_2$;

Step 8 for each particle of the set M_1, update the position $\vec{y}_i$ and velocity $\vec{v}_i$ as follows:

for j=1:n **do**

$v_{i,j} = |randn|(ybest_{i,j} - y_{i,j}) + |Randn|(lbest_{i,j} - y_{i,j})$;

$y_{i,j} = y_{i,j} + v_{i,j}$;

end for

Note that the above update is executed for the first particle $\vec{y}_1$ with a probability of 0.85.

Step 9 calculate $f(\vec{y}_i)$ and $G(\vec{y}_i)$, $i = 1, 2, \ldots, N_s$. Compare $\vec{y}_i$ and $\overrightarrow{ybest}_i$ using the feasibility-based rule, if $\vec{y}_i$ wins, then update $\overrightarrow{ybest}_i$;

Step 10 $P' = P' \cup M\_1$ and $Pbest' = Pbest' \cup M\_2$;

Step 11 $k = k + 1$;

Step 12 **end while**

Step 13 $P' = P' \cup P$ and $Pbest' = Pbest' \cup Pbest$;

Output: P' and $Pbest'$

Fig. 2 Pseudocode of swarm's splitting and sub-swarms' evolution

ity of PSO might not be very good since all particles should adjust their trajectory towards the local best. Based on the above consideration, $(N_s - 1)$ particles with the largest Euclidean distance from the local best are assigned as the other members of the first sub-swarm. Subsequently, these N_s particles are deleted from the swarm. Next, the first particle of the remaining swarm is assigned as the local best of the second sub-swarm, the $(N_s - 1)$ particles with the largest Euclidean distance from the local best are chosen as the other members of the second sub-swarm, and these N_s particles are eliminated from the swarm. The above process is executed constantly until the swarm is divided in to $\lfloor N/N_s \rfloor$ sub-swarms. It is necessary to note that Liang and Suganthan [9] adopted a similar idea to assign particles for a sub-swarm. In their method, the best and $(sn-1)$ worst particles are selected for each sub-swarm where sn is the size of the sub-swarm.

After all the sub-swarms are built, Eqs. (13) and (14) are exploited to evolve each sub-swarm. In each sub-swarm, since the local best is the fittest particle, other particles within the same sub-swarm can be made to follow it, which allows particles within the sub-swarm to be attracted to positions that make them even fitter. Moreover, since the sub-swarms are formed around different optima in parallel, the local bests provide the right guidance for particles in different sub-swarms to locate multiple optima, which can maintain the diversity of the swarm effectively.

It is necessary to note that if the local best of the sub-swarm is updated too frequently, the convergence speed of the sub-swarm will be relatively fast. Under this circumstance, once a "super" particle, which is much better than other particles in the sub-swarm, has been found, the sub-swarm might wander deeply into the neighborhood of such particle. Thus, the swarm might be trapped in a local optimum, due to the inability to adjust the velocity to continue the search at a finer grain. To address this situation, a simple yet effective mechanism is introduced: the local best of each sub-swarm undergoes the velocity and position updates with a probability of 0.85.

In this paper, if the variable value $x_{i,j}^{t+1}$ of particle $\vec{x}_i^{t+1}$ generated by Eqs. (13) and (14) violates the boundary constraint, the violated variable value is reflected back from the violated boundary using the following rule:

$$x_{i,j}^{t+1} = \begin{cases} 0.5(x_{i,j}^t + L_j), & \text{if} \quad x_{i,j}^t < L_j, \\ 0.5(x_{i,j}^t + U_j), & \text{if} \quad x_{i,j}^t > U_j. \end{cases} \tag{15}$$

Algorithm: updating the personal best by DE

Input: $Pbest = \{\overrightarrow{Pbest}_1, \ldots, \overrightarrow{Pbest}_N\}$

Step 1 **for** $i = 1{:}N$ **do**

Step 2 randomly select three different numbers r_1, r_2 and r_3 from $\{1, 2, \ldots, N\}\backslash i$

Step 3 $\vec{v}_i = \overrightarrow{Pbest}_{r1} + F(\overrightarrow{Pbest}_{r2} - \overrightarrow{Pbest}_{r3})$

Step 4 **for** j=1:n **do**

Step 5 $$u_{i,j} = \begin{cases} v_{i,j} & \text{if } rand_j \leqslant C_r \text{ or } j = j_{rand}, \\ x_{i,j} & \text{otherwise}, \end{cases}$$

Step 6 **end for**

Step 7 compare $\vec{u}_i$ with $\overrightarrow{Pbest}_i$ based on the feasibility-based rule, if $\vec{u}_i$ wins, then update $\overrightarrow{Pbest}_i$

Step 8 **end for**

Output: $Pbest$

Fig. 3 Pseudocode of updating the personal best by DE

3.2 Updating the personal best by DE

In order to further improve the global search capability of the above multi-swarm PSO, DE is incorporated in the following manner: DE is applied to the personal best $\overrightarrow{Pbest}_i^t$ of each particle $\vec{x}_i^t$. As a result, a corresponding trail vector $\vec{u}_i^t$ is yielded. Then, $\overrightarrow{Pbest}_i^t$ is compared with $\vec{u}_i^t$ and $\overrightarrow{Pbest}_i^t$ is updated if $\vec{u}_i^t$ wins. The above process is shown in Fig. 3. Note that, Munoz-Zavala et al. [15] also exploited a similar idea to perturb the personal best of each particle; however, only the mutation operator in DE is used.

It is interesting to note that by altering the memory of the personal best of each particle, the overall quality of the personal best will be improved, which can effectively enhance the performance of the swarm since the swarm will follow the improved personal best to fly at the subsequent iterations. In addition, DE can help the multi-swarm PSO to search a more promising region which might not be reached by the multi-swarm PSO during the evolution.

In our method, if the variable value $u_{i,j}^t$ of the trial vector $\vec{u}_i^t$ violates the boundary constraint, the violated variable value is either reflected back from the violated boundary or set to the boundary value using the following rule [21]:

$$u_{i,j}^t = \begin{cases} L_j, & \text{if} \quad (p \leqslant 0.5) \wedge (u_{i,j}^t < L_j), \\ U_j, & \text{if} \quad (p \leqslant 0.5) \wedge (u_{i,j}^t > U_j), \\ 2L_j - u_{i,j}^t, & \text{if} \quad (p > 0.5) \wedge (u_{i,j}^t < L_j), \\ 2U_j - u_{i,j}^t, & \text{if} \quad (p > 0.5) \wedge (u_{i,j}^t > U_j), \end{cases} \tag{16}$$

where p is a uniformly distributed random number between 0 and 1.

4 Experimental study

4.1 Benchmark test functions

An extensive empirical study is performed to investigate the performance of HMPSO. 13 benchmark test functions chosen from [22] are used. Information on these test functions is tabulated in Table 1. As shown in Table 1, these test functions include various types (linear, nonlinear, polynomial, quadratic and cubic) of objective functions with different numbers of decision variables (n) and a range of types (linear inequalities (LI), nonlinear inequalities (NI), linear equalities (LE), and nonlinear equalities (NE)) and number of constraints. In Table 1, a is the number of constraints active at the optimal solution, and $f(x^*)$ is the objective function value of the best known solution. In addition, $\rho = |\Omega|/|S|$ is the estimated ratio between the feasible region and the search space, where $|S|$ is the number of solutions randomly generated from S, $|\Omega|$ is the number of feasible solutions out of these $|S|$ solutions. In our experimental setup, $|S|$=1000000.

4.2 Parameter settings

In our experiments, the swarm size N is set to 60, and the size of each sub-swarm (N_s) is set to 8. For DE, the parameters are set to $F = 0.7$ and $C_r = 1.0$. It is noteworthy that there is no need to tune parameters for the local version of Krohling and Coelho's PSO used in this paper. The tolerance value ε for the equality constraints is set to 0.0001. The number of generations is set to 2500, thus the number of fitness function evaluations (FFEs) is equal to 300000. Each experiment is independently run 30 times in MATLAB (the source code may be obtained from the authors upon request),

Table 1 Details of 13 benchmark test functions

function	n	type of objective function	ρ/%	LI	NI	LE	NE	a	$f(x^*)$
$g01$	13	quadratic	0.0111	9	0	0	0	6	-15.000
$g02$	20	nonlinear	99.9971	0	2	0	0	1	-0.803619
$g03$	10	polynomial	0.0000	0	0	0	1	1	-1.000
$g04$	5	quadratic	51.1230	0	6	0	0	2	-30665.539
$g05$	4	cubic	0.0000	2	0	0	3	3	5126.498
$g06$	2	cubic	0.0066	0	2	0	0	2	-6961.814
$g07$	10	quadratic	0.0003	3	5	0	0	6	24.306
$g08$	2	nonlinear	0.8560	0	2	0	0	0	-0.095825
$g09$	7	polynomial	0.5121	0	4	0	0	2	680.630
$g10$	8	linear	0.0010	3	3	0	0	0	7049.248
$g11$	2	quadratic	0.0000	0	0	0	1	1	0.75
$g12$	3	quadratic	4.7713	0	1	0	0	0	-1.000
g13	5	nonlinear	0.0000	0	0	0	3	3	0.053950

Table 2 Experimental results obtained by HMPSO for 13 benchmark test functions over 30 independent runs

function	optimal	best	median	mean	worst	std
$g01$	-15.000	-15.000	-15.000	-15.000	-15.000	0.0e+00
$g02$	-0.803619	-0.803619	-0.803617	-0.798110	-0.776242	8.1e-03
$g03$	-1.000	-1.0005	-1.0005	-1.0005	-1.0005	3.8e-06
$g04$	-30665.539	-30665.5386718	-30665.5386718	-30665.5386718	-30665.5386718	1.1e-11
$g05$	5126.498	5126.496714	5126.496714	5126.496714	5126.496714	1.1e-08
$g06$	-6961.814	-6961.8138756	-6961.8138756	-6961.8138756	-6961.8138756	1.9e-12
$g07$	24.306	24.306209	24.306209	24.306209	24.306209	9.7e-14
$g08$	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825	2.4e-17
$g09$	680.630	680.6300574	680.6300574	680.6300574	680.6300574	4.5e-13
$g10$	7049.248	7049.2480205	7049.2480205	7049.2480205	7049.2480205	8.1e-12
$g11$	0.750	0.749900	0.749900	0.749900	0.749900	1.1e-16
$g12$	-1.000	-1.000	-1.000	-1.000	-1.000	0.0e+00
$g13$	0.053950	0.0539415	0.438803	0.342101	0.438960	1.5e-01

and each run is terminated only when the maximum number of generations has been elapsed.

4.3 Experimental results

Table 2 summarizes the experimental results using the above parameter settings. The table shows the known "optimal" solution for each test function and the best, median, mean,worst, and standard deviation of the objective function values derived from HMPSO over 30 runs.

As shown in Table 2, for each test function, the best solution is almost equivalent to the optimal solution. HMPSO is able to consistently find the global optima in 8 test functions (i.e., test functions $g01$, $g04$, $g06$, $g07$, $g08$, $g09$, $g10$, and $g12$). With respect to test functions with equality constraints (i.e., test functions $g03$, $g05$, $g11$, and $g13$) objective function values that are better than the optimal values have been found, it is because equality constraints are transformed into inequality constraints and relaxed by using the parameter ε. For test function $g02$, the near-optimal solutions are found in 18 runs and the exception occurs for 12 runs. It is important to note that the mean objective function value provided by HMPSO is quite close to the optimal value for this test function. For test function $g13$, the optimal solution is not consistently found. The main characteristic of this test function is three nonlinear equality constraints. For each test function, HMPSO is capable of finding feasible solutions in all runs.

4.4 Compare with the α constrained simplex method [23], the hybrid constrained optimization EA [6] and the improved stochastic ranking [24]

To verify the effectiveness of HMPSO, the solutions derived from HMPSO are compared with those provided by the α constrained simplex method (abbreviated to α Simplex)

[23], the hybrid constrained optimization EA (abbreviated to HCOEA) [6] and the improved stochastic ranking (abbreviated to ISR) [24]. In the α Simplex method, 30 independent runs are performed, all the equality constraints are relaxed using $\varepsilon = 0.0001$ and the maximum number of FFEs is about 30000 for test function $g12$ and about from 290000 to 330000 for the other test functions. For the HCOEA method, 30 independent runs are performed, the number of FFEs is equal to 240000, and all the equality constraints are relaxed using $\varepsilon = 1\mathrm{e}-07$. For the ISR method, 100 independent runs are performed using (60,400)-evolutionary strategy and 350000 FFEs.

As shown in Tables 3–5, the performance of HMPSO is compared in detail with α Simplex, HCOEA and ISR using the selected performance metrics, i.e., the best, mean and worst results. The results of α Simplex, HCOEA and ISR are directly taken from [23], [6] and [24]. It is evident that α Simplex, HCOEA and ISR provide very high-quality results for the 13 benchmark test functions. In Tables 3–5, the better results between the two compared algorithms are highlighted using boldface.

Compared with HMPSO, α Simplex finds a similar "best" result and better "mean" and "worst" results for test function $g13$. For test function $g02$, a similar "best" result is found by HMPSO and α Simplex. Nerverthless, HMPSO provides "mean" and "worst" results of a higher quality. HMPSO and α Simplex have similar performanc on the remaining 11 test functions. With respect to HCOEA, HMPSO finds a similar "best" result and better "mean" and "worst" results for test function $g07$. HMPSO surpasses HCOEA for test function $g10$ in terms of all performance metrics. In addition, HMPSO finds a better "best" result for test function

Table 3 Comparing HMPSO with respect to α Simplex [23] on 13 benchmark test functions

function	optimal	best result		mean result		worst result	
		HMPSO	α Simplex	HMPSO	α Simplex	HMPSO	α Simplex
$g01$	−15.000	−15.000	−15.000	−15.000	−15.000	−15.000	−15.000
$g02$	−0.803619	−0.803619	−0.803619	**−0.798110**	−0.784187	**−0.776242**	−0.754259
$g03$	−1.000	−1.001	−1.001	−1.001	−1.001	−1.001	−1.001
$g04$	−30665.539	−30665.539	−30665.539	−30665.539	−30665.539	−30665.539	−30665.539
$g05$	5126.498	5126.497	5126.497	5126.497	5126.497	5126.497	5126.497
$g06$	−6961.814	−6961.814	−6961.814	−6961.814	−6961.814	−6961.814	−6961.814
$g07$	24.306	24.306	24.306	24.306	24.306	24.306	24.306
$g08$	−0.095825	−0.095825	−0.095825	−0.095825	−0.095825	−0.095825	−0.095825
$g09$	680.630	680.630	680.630	680.630	680.630	680.630	680.630
$g10$	7049.248	7049.248	7049.248	7049.248	7049.248	7049.248	7049.248
$g11$	0.750	0.750	0.750	0.750	0.750	0.750	0.750
$g12$	−1.000	−1.000	−1.000	−1.000	−1.000	−1.000	−1.000
$g13$	0.053950	0.053942	0.053942	0.342101	**0.066770**	0.438960	**0.438803**

Table 4 Comparing HMPSO with respect to HCOEA [6] on 13 benchmark test functions

function	optimal	best result		mean result		worst result	
		HMPSO	HCOEA	HMPSO	HCOEA	HMPSO	HCOEA
$g01$	−15.000	−15.000	−15.000	−15.000	−15.000	−15.000	−15.000
$g02$	−0.803619	**−0.803619**	−0.803241	−0.798110	**−0.801258**	−0.776242	**−0.792363**
$g03$	−1.000	−1.001	−1.000	−1.001	−1.000	−1.001	−1.000
$g04$	−30665.539	−30665.539	−30665.539	−30665.539	−30665.539	−30665.539	−30665.539
$g05$	5126.498	5126.497	5126.498	5126.497	5126.498	5126.497	5126.498
$g06$	−6961.814	−6961.814	−6961.814	−6961.814	−6961.814	−6961.814	−6961.814
$g07$	24.306	24.306	24.306	**24.306**	24.307	**24.306**	24.309
$g08$	−0.095825	−0.095825	−0.095825	−0.095825	−0.095825	−0.095825	−0.095825
$g09$	680.630	680.630	680.630	680.630	680.630	680.630	680.630
$g10$	7049.248	**7049.248**	7049.287	**7049.248**	7049.525	**7049.248**	7049.984
$g11$	0.750	0.750	0.750	0.750	0.750	0.750	0.750
$g12$	−1.000	−1.000	−1.000	−1.000	−1.000	−1.000	−1.000
$g13$	0.053950	0.053942	0.053950	0.342101	**0.053950**	0.438960	**0.053950**

Table 5 Comparing HMPSO with respect to ISR [24] on 13 benchmark test functions

function	optimal	best result		mean result		worst result	
		HMPSO	ISR	HMPSO	ISR	HMPSO	ISR
$g01$	-15.000	-15.000	-15.000	-15.000	-15.000	-15.000	-15.000
$g02$	-0.803619	-0.803619	-0.803619	**-0.798110**	-0.772078	**-0.776242**	-0.683055
$g03$	-1.000	-1.001	-1.001	-1.001	-1.001	-1.001	-1.001
$g04$	-30665.539	-30665.539	-30665.539	-30665.539	-30665.539	-30665.539	-30665.539
$g05$	5126.498	5126.497	5126.498	5126.497	5126.498	5126.497	5126.498
$g06$	-6961.814	-6961.814	-6961.814	-6961.814	-6961.814	-6961.814	-6961.814
$g07$	24.306	24.306	24.306	24.306	24.306	**24.306**	24.308
$g08$	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825
$g09$	680.630	680.630	680.630	680.630	680.630	680.630	680.630
$g10$	7049.248	7049.248	7049.248	**7049.248**	7049.249	**7049.248**	7049.296
$g11$	0.750	0.750	0.750	0.750	0.750	0.750	0.750
$g12$	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000
$g13$	0.053950	0.053942	0.053942	0.342101	**0.096276**	0.438960	**0.438803**

$g02$ which is just the global optimum of this test function. HCOEA provides better "mean" and "worst" results for test functions $g02$ and $g13$. The performance of HMPSO and HCOEA are silimar for the rest of test functions. In comparison with ISR, HMPSO provides better "mean" and "worst" results for test functions $g02$ and $g10$. Moreover, the "worst" result derived from HMPSO is better than that of ISR for test function $g07$. ISR obtains better "mean" and "worst" results than HMPSO for test function $g13$. HMPSO and ISR have marginal performanc difference on the remaining 9 test functions.

From the above discussion, one can conclude that the performance of HMPSO is very competitive with that of α Simplex, HCOEA and ISR which are the state-of-the-art methods in the area of constrained evolutionary optimization. It is important to emphasize that HMPSO and HCOEA have the similar evolutionary framework (i.e., global search plus local search) as discussed in Section 1.

However, HCOEA contains a problem-dependent crossover operator, i.e., the scaling factor of the crossover operator should be tuned acorrding to the problems at hand, which limits the real-world application of HCOEA. The above drawback has been overcome in this paper by using multi-swarm PSO as the search algorithm.

5 Discussion

5.1 Effectiveness of the multi-swarm PSO

This paper adopts the multi-swarm PSO to search the global optima of the test functions. In the multi-swarm PSO, the local version of Krohling and Coelho's PSO is used. In order to examine the effectiveness of the multi-swarm PSO, another algorithm (denoted as HMPSO_1) is performed in which the local version of Krohling and Coelho's PSO is replaced with the global version of Krohling and Coelho's PSO, as a result, there exists only one swarm in the algorithm. In HMPSO_1, 30 independent runs are executed and the same number of FFEs with HMPSO is maintained to have a fair comparison.

Based on the observation, the performance of HMPSO_1 only has a significant difference with that of HMPSO on test functions $g01$, $g02$, $g05$ and $g13$. Therefore, Table 6 only summarizes the experimental results of HMPSO and HMPSO_1 for these four test functions in terms of the following performance metrics: the best, mean, worst and standard deviations of the objective function values. From Table 6, it is clear that HMPSO performs better than HMPSO_1 on all performance metrics for test function $g02$. In addition, the degradation of performance tends to take place in test functions $g01$ and $g05$ for HMPSO_1, since HMPSO_1 is unable to consistently converge to the global optima for these two test functions. This may be because the diversity of the swarm is not very good. Although HMPSO and HMPSO_1 cannot succeed to solve test function $g13$ in all runs, HMPSO provides better "mean" and "worst" results for this test function than HMPSO_1.

The above discussion verifies that the multi-swarm PSO is a more effective technique than the PSO with only one swarm for solving COPs.

5.2 Effectiveness of updating the personal best by DE

The effectiveness of updating the personal best by DE has been inverstigated by another algorithm (denoted as HMPSO_2) in which the operation of updating the personal best by DE has been removed. This time, 30 runs are performed. In order to have a fair comparison, the number of

FFEs is not changed.

Since the performance difference between HMPSO and HMPSO_2 is remarkable only for test functions $g02$, $g03$, $g05$, $g07$, $g09$, $g10$, $g11$, and $g13$, Table 7 only summarizes the experimental results of HMPSO and HMPSO_2 for these 8 test functions in terms of the performance criteia used in Section 5.1. As shown in Table 7, HMPSO outperforms HMPSO_2 for test functions $g02$, $g03$, $g07$, $g09$ and $g10$ with regard to all performance criteria. Compared with HMPSO_2, HMPSO provides a similiar "best" result and better "mean" and "worst" results for test function $g11$. In addtion, for test functions $g05$ and $g11$ HMPSO_2 cannot consistenly enter the feasible region. More importantly, for test function $g05$, feasible solution cannot be found in any run.

The above discussion signifies that the operation of updating the personal best by DE is capable of enhancing the global search ability of HMPSO significantly.

5.3 Effect of the parameter C_r in DE

In the proposed HMPSO, the crossover control parameter C_r of DE is set to 1.0, which means the trial vector is totally equal to the mutant vector, and that the target vector does not contribute any information for the trial vector. In order to verify the effectiveness of the above choice, we test DE with five different C_r: 0.6, 0.7, 0.8, 0.9 and 1.0. For each setting, 30 independent runs are performed. Table 8 summarizes the mean of the objective funciton values.

As decipted in Table 8, while in the case of $C_r = 1.0$ the mean result of test funciton $g02$ is worse than other results;

Table 6 Comparison of HMPSO with HMPSO_1 on test functions $g01$, $g02$, $g05$ and $g13$ over 30 runs

function	optimal	method	best	mean	worst	st. dev
$g01$	-15.000	HMPSO	-15.000	-15.000	-15.000	0.0e+00
		HMPSO_1	-15.000	-14.630	-12.453	8.5e-01
$g02$	-0.803619	HMPSO	-0.803619	-0.798110	-0.776242	8.1e-03
		HMPSO_1	-0.728023	-0.486414	-0.379468	8.0e-02
$g05$	5126.498	HMPSO	5126.497	5126.497	5126.497	1.1e-08
		HMPSO_1	5126.497	5129.203	5154.975	7.6e+00
$g13$	0.053950	HMPSO	0.0539415	0.342101	0.438960	1.5e-01
		HMPSO_1	0.0539415	0.401918	1.000000	2.7e-01

Table 7 Comparison of HMPSO with HMPSO_2 on test functions $g02$, $g03$, $g05$, $g07$, $g09$, $g10$, $g11$, and $g13$ over 30 runs; (#) denotes the number that feasible solutions are found in the final population over 30 trials

function	optimal	method	best	mean	worst	st. dev
$g02$	-0.803619	HMPSO	-0.803619	-0.798110	-0.776242	8.1e-03
		HMPSO_2	-0.783902	-0.772573	-0.748446	8.5e-03
$g03$	-1.000	HMPSO	-1.001	-1.001	-1.001	3.8e-06
		HMPSO_2	-0.757	-0.566	-0.334	1.1e-01
$g05$	5126.498	HMPSO	5126.497	5126.497	5126.497	1.1e-08
		HMPSO_2		(0)		
$g07$	24.306	HMPSO	24.306	24.306	24.306	9.7e-14
		HMPSO_2	24.799	25.060	25.511	1.8e-01
$g09$	680.630	HMPSO	680.630	680.630	680.630	4.5e-13
		HMPSO_2	680.654	680.676	680.710	1.6e-02
$g10$	7049.248	HMPSO	7049.248	7049.248	7049.248	8.1e-12
		HMPSO_2	7212.536	7332.059	7402.391	5.7e+01
$g11$	0.750	HMPSO	0.750	0.750	0.750	1.1e-16
		HMPSO_2	0.750	0.784	0.861	3.3e-02
$g13$	0.053950	HMPSO	0.0539415	0.342101	0.438960	1.5e-01
		HMPSO_2		(6)		

Table 8 Experimental results on 13 benchmark functions with varying crossover control parameter; 30 independent runs are performed; (#) denotes the number that feasible solutions are found in the final population in 30 trials

function	0.6	0.7	0.8	0.9	1.0
$g01$	-15.000	-15.000	-15.000	-15.000	-15.000
$g02$	-0.802470	-0.802242	-0.802698	-0.803171	-0.798110
$g03$	-0.452	-0.458	-0.652	-0.931	-1.001
$g04$	-30665.539	-30665.539	-30665.539	-30665.539	-30665.539
$g05$	5194.493	5185.872	5144.279	5127.689	5126.497
$g06$	-6961.814	-6961.814	-6961.814	-6961.814	-6961.814
$g07$	24.427	24.356	24.310	24.306	24.306
$g08$	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825
$g09$	680.630	680.630	680.630	680.630	680.630
$g10$	7101.719	7059.066	7049.661	7049.248	7049.248
$g11$	0.750	0.750	0.750	0.750	0.750
$g12$	-1.000	-1.000	-1.000	-1.000	-1.000
$g13$	(27)	0.961552	0.849426	0.556571	0.342101

the mean results provided by $C_r = 1.0$ are much better than other results for test functions $g03$, $g05$ and $g13$. Besides, in the case of $C_r = 1.0$ the mean results of test functions $g07$ and $g10$ are similar to those of $C_r = 0.9$, and of much higher quality than those of C_r =0.6, 0.7 and 0.8. It is necessary to note that the algorithm cannot consistently find feasible solutions for test function $g13$ when $C_r = 0.6$. In general, the overall performance of $C_r = 1.0$ is better than that of other settings for C_r. It may be because the diversity of the population is very good when $C_r = 1.0$ since in this case the information provided by the targer vector is not utilized and the trial vector is quite different from its target vector.

Based on the above discussion, it is clear that $C_r = 1.0$ is a reasonable choice for HMPSO.

5.4 Benchmark test functions collected in the 2006 IEEE Congress on Evolutionary Computation (CEC2006)

The performance of HMPSO has been further investigated by taking advantage of 24 benchmark test functions collected in the CEC2006 [25], which include the 13 benchmark test functions used in Section 4. In our experiments, each test function is run for 25 independent trials with 500000 FFEs according to [25]. The other parameters are not changed compared with those in Section 4. For each trial, the function error value $(f(\vec{x}) - f(\vec{x}^*))$ for the achieved best solutin $\vec{x}$ after 5×10^3 FFEs, 5×10^4 FFEs, and 5×10^5 FFEs is recorded respectively, where $\vec{x}^*$ denotes the best known solution and $f(\vec{x}^*)$ denotes the objective function value of the best known solution. The best, median, worst, mean and standard deviation of the function error values for the 25 runs are reported in Tables 9–12. c denotes the number of violated constraints (including the number of constraint violations more than 1, 0.01, and 0.0001, respectively) at the median solution, and $\bar{v}$ is the mean value of the violations of all constraints at the median solution. The numbers in the parenthesis after the function error values indicate the number of violated constraints at the corresponding values. In addition, the best, median, worst, mean, and standard deviation of the number of FFEs to achieve the fixed accuracy level $(f(\vec{x}) - f(\vec{x}^*)) < 0.0001$ are summarized in Table 13. Meanwhile, the feasible rate (rate of runs during which at least one feasible solution is found), the success rate (rate of runs during which the algorithm finds a feasible solution satisfying the required accuracy) and the success performance (the mean number of FFEs of the successful runs divided by the success rate) are also shown in Table 13. It is necessary to note that successful run means the algorithm finds a feasible solution $\vec{x}$ satisfying $(f(\vec{x}) - f(\vec{x}^*)) < 0.0001$. A new optimal solution is found in this paper for test function $g17$, i.e., $\vec{x}^*$= (201.784462493550, 100, 383.071034852773, 420, -10.907677947103, 0.073148231208) with $f(\vec{x}^*)$= 8853.53387480648, therefore the best known solution reported in Ref. [25] is replaced with the new optimal solution provided in this paper.

As shown in Tables 9–12, HMPSO is able to consistently enter the feasible region with 5×10^3 FFEs, 5×10^4 FFEs and 5×10^5 FFEs for 13 test functions, 18 test functions and 21 test functions, respectively. For test function $g21$, HMPSO can find feasible solutions with 5×10^5 FFEs for the majority of runs. While HMPSO is unable to find feasible solutions for test functions $g20$ and $g22$ with 5×10^5 FFEs, the resulting solutions found by HMPSO are very close to the feasible region for test function $g20$ since the mean constraint violation of the median solution is only 1.3818e-02. Additionally, the resulting solutions derived from HMPSO can achieve the fixed accuracy level for 22 test functions with the exception

Table 9 Error values achieved when FFEs= 5×10^3, FFEs= 5×10^4, and FFEs= 5×10^5 for test functions 1–6

FFEs		$g01$	$g02$	$g03$	$g04$	$g05$	$g06$
	best	2.5268e+00 (0)	4.2979e-01 (0)	7.0391e-01 (0)	9.2699e+00 (0)	2.3614e+02 (3)	4.1530e+00 (0)
	median	4.2245e+00 (0)	5.1255e-01 (0)	9.4373e-01 (0)	3.5971e+01 (0)	8.5080e+00 (3)	2.2591e+01 (0)
	worst	5.6653e+00 (0)	5.6804e-01 (0)	9.9402e-01 (1)	6.5143e+01 (0)	2.8299e+02 (3)	1.4634e+02 (0)
5×10^3	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 3, 0	0, 0, 0
	$\bar{v}$	0	0	0	0	2.0470e-01	0
	mean	4.1303e+00	5.1317e-01	8.9645e-01	3.5850e+01	1.2840e+02	3.4555e+01
	std	8.3860e-01	3.1003e-02	9.2915e-02	1.4375e+01	1.2297e+02	3.2293e+01
	best	3.1888e-04 (0)	7.6094e-02 (0)	2.7898e-01 (0)	7.6397e-11 (0)	2.4736e-01 (0)	3.3651e-11 (0)
	median	1.4687e-03 (0)	1.7246e-01 (0)	6.1993e-01 (0)	8.3673e-11 (0)	1.0985e+02 (0)	3.3651e-11 (0)
	worst	4.0823e-03 (0)	2.5164e-01 (0)	8.8787e-01 (0)	1.1641e-10 (0)	4.4455e+02 (0)	3.3651e-11 (0)
5×10^4	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	$\bar{v}$	0	0	0	0	0	0
	mean	1.7785e-03	1.7323e-01	6.0604e-01	8.4401e-11	1.3133e+02	3.3651e-11
	std	1.2336e-03	4.7686e-02	1.6734e-01	9.1553e-12	1.1877e+02	1.3191e-26
	best	0.0000e+00 (0)	2.0213e-08 (0)	−1.0002e-11 (0)	7.6397e-11 (0)	−1.8189e-12 (0)	3.3651e-11 (0)
	median	0.0000e+00 (0)	1.9264e-07 (0)	−1.0002e-11 (0)	7.6397e-11 (0)	−1.8189e-12 (0)	3.3651e-11 (0)
	worst	0.0000e+00 (0)	2.5775e-02 (0)	−1.0002e-11 (0)	7.6397e-11 (0)	−1.8189e-12 (0)	3.3651e-11 (0)
5×10^5	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	$\bar{v}$	0	0	0	0	0	0
	mean	0.0000e+00	5.3901e-03	−1.0002e-11	7.6397e-11	−1.8189e-12	3.3651e-11
	std	0.0000e+00	7.0184e-03	2.1259e-16	2.6382e-26	1.2366e-27	1.3191e-26

Table 10 Error values achieved when FFEs= 5×10^3, FFEs= 5×10^4, and FFEs= 5×10^5 for test functions 7–12

FFEs		$g07$	$g08$	$g09$	$g10$	$g11$	$g12$
	best	2.7810e+01 (0)	8.2459e-11 (0)	1.9191e+01 (0)	3.0022e+03 (0)	4.2168e-07 (0)	6.3890e-05 (0)
	median	5.8291e+01 (0)	9.0829e-11 (0)	2.8108e+01 (0)	4.0790e+03 (0)	1.8671e-01 (0)	3.2680e-04 (0)
	worst	1.4631e+02 (0)	2.4411e-10 (0)	6.1193e+01 (0)	6.9721e+03 (0)	2.4810e-01 (0)	6.6592e-03 (0)
5×10^3	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	$\bar{v}$	0	0	0	0	0	0
	mean	6.2413e+01	9.9205e-11	3.2350e+01	4.2917e+03	1.4220e-01	7.9423e-04
	std	2.2952e+01	3.3064e-11	1.2236e+01	9.1180e+02	8.8459e-02	1.7068e-03
	best	3.8955e-02 (0)	8.1964e-11 (0)	5.7706e-07 (0)	7.1124e+00 (0)	0.0000e+00 (0)	0.0000e+00 (0)
	median	1.1788e-01 (0)	8.1964e-11 (0)	1.5215e-06 (0)	1.3035e+01 (0)	0.0000e+00 (0)	0.0000e+00 (0)
	worst	2.5677e-01 (0)	8.1964e-11 (0)	7.0542e-06 (0)	3.0066e+01 (0)	3.3417e-14 (0)	0.0000e+00 (0)
5×10^4	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	$\bar{v}$	0	0	0	0	0	0
	mean	1.2542e-01	8.1964e-11	1.9282e-06	1.6151e+01	1.3589e-15	0.0000e+00 (0)
	std	4.8960e-02	5.6655e-18	1.3604e-06	6.8836e+00	6.6798e-15	0.0000e+00 (0)
	best	7.9758e-11 (0)	8.1964e-11 (0)	−9.8225e-11 (0)	6.2755e-11 (0)	0.0000e+00 (0)	0.0000e+00 (0)
	median	7.9779e-11 (0)	8.1964e-11 (0)	−9.8111e-11 (0)	6.2755e-11 (0)	0.0000e+00 (0)	0.0000e+00 (0)
	worst	7.9790e-11 (0)	8.1964e-11 (0)	−9.8111e-11 (0)	6.3664e-11 (0)	0.0000e+00 (0)	0.0000e+00 (0)
5×10^5	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	$\bar{v}$	0	0	0	0	0	0
	mean	7.9777e-11	8.1964e-11	−9.8134e-11	6.2791e-11	0.0000e+00	0.0000e+00
	std	8.6489e-15	5.6655e-18	4.6412e-14	1.8189e-13	0.0000e+00	0.0000e+00

Table 11 Error values achieved when FFEs= 5×10^3, FFEs= 5×10^4, and FFEs= 5×10^5 for test functions 13–18

FFEs		$g13$	$g14$	$g15$	$g16$	$g17$	$g18$
	best	9.4285e-01 (3)	6.0543e+00 (3)	1.2942e-01 (2)	3.4862e-02 (0)	1.0268e+02 (4)	4.6194e-01 (1)
	median	9.5123e-01 (3)	−4.2953e+00 (3)	6.7195e-01 (2)	6.2033e-02 (0)	1.2548e+02 (4)	7.9413e-01 (3)
	worst	8.4279e-01 (3)	−2.2049e+01 (3)	1.0503e+00 (2)	9.6150e-02 (0)	3.2548e+02 (4)	9.6708e-01 (4)
5×10^3	c	0, 3, 0	0, 3, 0	0, 1, 1	0, 0, 0	3, 1, 0	0, 3, 0
	$\bar{v}$	1.2916e-01	2.4966e-01	8.0970e-02	0	1.8988e+00	5.2045e-02
	mean	8.5233e-01	−2.6751e+00	9.8792e-01	6.3904e-02	1.4191e+02	7.6526e-01
	std	3.4058e-01	7.5738e+00	1.3403e+00	1.8453e-02	1.2297e+02	2.0279e-01
	best	4.2680e-01 (0)	1.6882e-03 (0)	6.0940e-11 (0)	1.8286e-09 (0)	3.4784e+01 (0)	3.7269e-03 (0)
	median	8.9506e-01 (0)	4.2860e-03 (0)	2.7945e-01 (0)	4.7401e-09 (0)	9.6230e+01 (0)	7.9930e-03 (0)
	worst	6.0073e-01 (3)	1.8968e-02 (0)	4.2892e+00 (0)	1.3219e-08 (0)	9.0904e+01 (4)	2.0537e-02 (0)
5×10^4	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	$\bar{v}$	0	0	0	0	0	0
	mean	7.8259e-01	6.8354e-03	6.6810e-01	5.5710e-09	1.1021e+02	9.5024e-03
	std	1.8720e-01	5.4894e-03	1.0733e+00	3.0955e-09	7.4661e+01	4.3701e-03
	best	4.1900e-11 (0)	8.5123e-12 (0)	6.0822e-11 (0)	6.5213e-11 (0)	−1.8189e-12 (0)	1.5561e-11 (0)
	median	3.8486e-01 (0)	8.5123e-12 (0)	6.0822e-11 (0)	6.5213e-11 (0)	7.4057e+01 (0)	1.5561e-11 (0)
	worst	3.8486e-01 (0)	8.5194e-12 (0)	6.0822e-11 (0)	6.5213e-11 (0)	7.5028e+01 (0)	1.5561e-11 (0)
5×10^5	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	$\bar{v}$	0	0	0	0	0	0
	mean	2.6170e-01	8.5148e-12	6.0822e-11	6.5213e-11	5.9344e+01	1.5561e-11
	std	1.8323e-01	3.4809e-15	0.0000e+00	2.6382e-26	3.0285e+01	5.2857e-17

Table 12 Error values achieved when FFEs= 5×10^3, FFEs= 5×10^4, and FFEs= 5×10^5 for test functions 19–24

FFEs		$g19$	$g20$	$g21$	$g22$	$g23$	$g24$
	best	1.0702e+02 (0)	4.0596e+00 (20)	6.1019e+02 (4)	1.6001e+04 (19)	4.0005e+02 (0)	2.0313e-05 (0)
	median	1.5467e+02 (0)	3.6156e+00 (20)	5.7427e+02 (5)	1.9232e+04 (15)	1.7287e+02 (4)	8.3642e-05 (0)
	worst	2.2713e+02 (0)	7.2877e+00 (20)	−1.3505e+00 (4)	3.3570e+03 (20)	4.7763e+01 (5)	2.7550e-04 (0)
5×10^3	c	0, 0, 0	2, 15, 3	0, 5, 0	14, 1, 0	1, 3, 0	0, 0, 0
	$\bar{v}$	0	2.7966e+00	6.1364e-02	6.2343e+05	3.4280e-01	0
	mean	1.6431e+02	5.4908e+00	5.0113e+02	1.0370e+04	1.3406e+02	1.1274e-04
	std	3.4837e+01	1.2169e+00	2.5244e+02	6.1103e+03	3.7941e+02	7.0347e-05
	best	1.1102e+00 (0)	−3.5659e-02 (19)	2.0185e-04 (0)	6.9038e+03 (19)	5.4224e+01 (0)	4.6735e-12 (0)
	median	2.2984e+00 (0)	−3.5593e-02 (20)	1.3111e+02 (0)	4.1056e+03 (19)	3.7931e+02 (0)	4.6735e-12 (0)
	worst	4.4589e+00 (0)	−3.7925e-02 (20)	6.9729e+02 (4)	1.0356e+04 (19)	4.0494e+02 (2)	4.6735e-12 (0)
5×10^4	c	0, 0, 0	0, 14, 6	0, 0, 0	18, 1, 0	0, 0, 0	0, 0, 0
	$\bar{v}$	0	3.3732e-02	0	8.0304e+03	0	0
	mean	2.4427e+00	−3.2376e-02	2.2826e+02	1.0042e+04	3.3507e+02	4.6735e-12
	std	9.1299e-01	2.0785e-02	2.6930e+02	4.6137e+03	1.7708e+02	8.2445e-28
	best	4.9233e-11 (0)	−3.7261e-03 (15)	−3.7800e-10 (0)	1.4096e+04 (13)	−1.7053e-13 (0)	4.6735e-12 (0)
	median	1.1917e-10 (0)	−1.1965e-02 (17)	1.3097e+02 (0)	1.4760e+04 (13)	1.0800e-12 (0)	4.6735e-12 (0)
	worst	5.0407e-09 (0)	−2.9593e-02 (18)	6.2494e+02 (1)	5.5870e+03 (19)	5.7355e-11 (0)	4.6735e-12 (0)
5×10^5	c	0, 0, 0	0, 1, 10	0, 0, 0	6, 0, 4	0, 0, 0	0, 0, 0
	$\bar{v}$	0	1.3818e-02	0	5.5976e+00	0	0
	mean	3.5565e-10	−5.7598e-03	2.5325e+02	9.7792e+03	5.9640e-12	4.6735e-12
	std	9.8397e-10	1.4944e-02	2.9653e+02	4.1754e+03	1.4183e-11	8.2445e-28

Table 13 Number of FFEs to achieve the fixed accuracy level ($(f(\vec{x}) - f(\vec{x}^*)) \leqslant 0.0001$), success rate, feasible rate and success performance

prob.	best	median	worst	mean	std	feasible rate/%	success rate/%	success performance
$g01$	53075	60422	70294	60760.6	5188.1	100	100	60760.6
$g02$	134120	153230	189467	157092.3	15344.8	100	56	280521.9
$g03$	127332	194539	258747	194523.4	36607.1	100	100	194523.4
$g04$	20852	23017	25336	22994.2	1272.4	100	100	22994.2
$g05$	26621	172732	315697	165759.5	73911.1	100	100	165759.5
$g06$	11657	13614	14655	13561.9	689.9	100	100	13561.9
$g07$	101996	110418	120366	109835.1	3995.8	100	100	109835.1
$g08$	1214	1671	2011	1655.8	195.8	100	100	1655.8
$g09$	33965	37764	41682	37686.7	1717.6	100	100	37686.7
$g10$	138459	147287	159281	147622.9	5697.9	100	100	147622.9
$g11$	9707	20406	40256	22469.5	7289.5	100	100	22469.5
$g12$	754	6607	9595	6331.4	2105.5	100	100	6331.4
$g13$	308339	359499	421069	368631.5	37067.2	100	44	837798.8
$g14$	59828	66935	73282	67195.8	3471.6	100	100	67195.8
$g15$	19600	99269	192283	91441.4	52205.3	100	100	91441.4
$g16$	18234	20531	22381	20310.3	1073.9	100	100	20310.3
$g17$	274158	313490	342959	311528.6	31757.2	100	24	1298035.8
$g18$	92104	100892	118781	102022.1	6831.2	100	100	102022.1
$g19$	183294	217174	254184	218252.1	19005.7	100	100	218252.1
$g20$	—	—	—	—	—	0	0	—
$g21$	56592	63668	70296	64296	4413.1	68	32	200925
$g22$	—	—	—	—	—	0	0	—
$g23$	194461	239326	357890	252811	47203.5	100	100	252811
$g24$	4085	4774	5467	4789.3	359.6	100	100	4789.3

of test functions $g20$ and $g22$. Among these 22 test functions, HMPSO can succeed to solve 18 test functions in all the 25 runs.

From Table 13, it is clear that HMPSO is able to achieve 100% feasbile rate for 21 test functions except for test functions $g20$, $g21$ and $g22$. For test function $g21$, the feasible rate is 68%. In addition, HMPSO can attain 100% success rate for 18 test functions except for test functions $g02$, $g13$, $g17$, $g20$, $g21$ and $g22$. For test functions $g02$, $g13$, $g17$ and $g21$, the success rate is 56%, 44%, 24% and 32%, respectively. HMPSO cannot solve test functions $g20$ and $g22$ successfully.

The above discusssion indicates that HMPSO is very effective in sloving these 24 test functions.

6 Conclusion

This paper proposes a hybrid multi-swarm PSO (HMPSO) to solve COPs. The method proposed divides the swarm into several sub-swarms and each sub-swarm uses the local version of Krohling and Coelho's PSO as the search engine. In addition, DE is incorporated to update the personal best of each particle in order to guide the swarm toward a more promising region. A comparative study of the proposed HMPSO and three state-of-the-art methods on 13 benchmark test functions is presented. The simulation results show the proposed HMPSO is highly competitive and able to obtain high-quality solutions for most of the test functions. Moreover, the effectiveness of some mechanisms proposed in this paper has been verified by different experiments. Meanwhile, the effect of the parameter C_r in DE on the performance of HMPSO is also investigated. Afterward, HMPSO is applied to solve 24 benchmark test functions collected in the CEC2006. The experimental results show that HMPSO is able to solve 22 benchmark test functions. As part of our future work, we are considering the possibility of improving the constraint-handling technique (note that in this paper only the feasibility-based rule is used to compare the particles). In addition, this paper only uses classic DE to improve the performance of multi-swarm PSO, whether HMPSO could be further improved by making use of more advanced DE (such as DE in Refs. [26−29]) is another subject of our future work.

Acknowledgements This work was supported in part by the National Natural Science Foundation of China (Grant Nos. 60805027 and 90820302), in part by the Research Fund for the Doctoral Program of

Higher Education (Grant No. 200805330005), in part by Hunan S&T Funds (Grant No. 06IJY3035), and in part by the Graduate Innovation Foundation of Central South University (1343-74334000016).

References

1. Michalewicz Z, Schoenauer M. Evolutionary algorithm for constrained parameter optimization problems. Evolutionary Computation, 1996, 4(1): 1–32
2. Coello Coello C A. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Computer Methods in Applied Mechanics and Engineering, 2002, 191(11–12): 1245–1287
3. Mezura-Montes E, Coello Coello C A. A simple multimembered evolution strategy to solve constrained optimization problems. IEEE Transactions on Evolutionary Computation, 2005, 9(1): 1–17
4. Cai Z, Wang Y. A multiobjective optimization-based evolutionary algorithm for constrained optimization. IEEE Transactions on Evolutionary Computation, 2006, 10(6): 658–675
5. Wang Y, Cai Z, Zhou Y, Zeng W. An adaptive trade-off model for constrained evolutionary optimization. IEEE Transactions on Evolutionary Computation, 2008,12(1): 80–92
6. Wang Y, Cai Z, Guo G, Zhou Y. Multiobjective optimization and hybrid evolutionary algorithm to solve constrained optimization problems. IEEE Transactions on Systems Man and Cybernetics, Part B: Cybernetics, 2007, 37(3): 560–575
7. Wang Y, Liu H, Cai Z, Zhou Y. An orthogonal design based constrained evolutionary optimization algorithm. Engineering Optimization, 2007, 39 (6): 715–736
8. Kennedy J, Eberhart R. Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, 1995, 1942–1948
9. Liang J J, Suganthan P N. Dynamic multi-swarm particle swarm optimizer with a novel constraint-handling mechanism. In: Proceedings of the Congress on Evolutionary Computation (CEC'2006). IEEE Press, 2006, 9–16
10. Krohling R A, Coelho L S. Coevolutionary particle swarm optimization using Gaussian distribution for solving constrained optimization problems. IEEE Transactions on Systems Man and Cybernetics, Part B: Cybernetics, 2006, 36(6): 1407–1416
11. He Q, Wang L. An effective co-evolutionary particle swarm optimization for constrained engineering design problems. Engineering Application of Artificial Intelligence, 2007, 20(1): 89–99
12. He Q, Wang L. A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization. Applied Mathematics and Computation, 2007, 186(2): 1407–1422
13. Deb K. An efficient constraint handling method for genetic algorithms. Computer Methods in Applied Mechanics and Engineering, 2000, 18(2–4): 311–338
14. Pulido T G, Coello Coello C A. A constraint-handling mechanism for particle swarm optimization. In: Proceedings of 2004 Congress on Evolutionary Computation (CEC'2004). IEEE Press, 2004, 1396–1403
15. Munoz-Zavala A E, Hernandez-Aguirre A, Villa-Diharce E R, Botello-Rionda S. PESO+ for constrained optimization. In: Proceedings of the Congress on Evolutionary Computation (CEC'2006). IEEE Press, 2006, 231–238
16. Zielinski K, Laur R. Constrained single-objective optimization using particle swarm optimization. In: Proceedings of the Congress on Evolutionary Computation (CEC'2006). IEEE Press, 2006, 443–450
17. Parsopoulos K E, Vrahatis M N. On the computation of all global minimizers through particle swarm optimization. IEEE Transactions on Evolutionary Computation, 2004, 8(3): 211–224
18. Shi Y, Eberhart R C. A modified particle swarm optimizer. In: Proceedings of IEEE Conference on Evolutionary Computation, 1998, 69–73
19. Clerc M, Kennedy J. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation, 2002, 6(1): 58–73
20. Storn R, Price K. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. International Computer Science Institute, Berkeley, Technical Report TR-95-012, 1995
21. Brest J, Zumer V, Maucec M S. Self-adaptive differential evolution algorithm in constrained real-parameter optimization. In: Proceedings of the Congress on Evolutionary Computation (CEC'2006). IEEE Press, 2006, 215–222
22. Runarsson T P, Yao X. Stochastic ranking for constrained evolutionary optimization. IEEE Transactions on Evolutionary Computation, 2000, 4(3): 284–294
23. Takahama T, Sakai S. Constrained optimization by applying the α constrained method to the nonlinear simplex method with mutations. IEEE Transactions on Evolutionary Computation, 2005, 9(5): 437–451
24. Runarsson T P, Yao X. Search bias in constrained evolutionary optimization. IEEE Transactions on Systems, Man, and Cybernetics, Part C, 2005, 35(2): 233–243
25. Liang J J, Runarsson T P, Mezura-Montes E, Clerc M, Suganthan P N, Coello Coello C A, Deb K. Problem definitions and evaluation criteria for the CEC 2006. Special Session on Constrained Real-Parameter Optimization, Technical Report. Singapore Nanyang Technological University, 2006
26. Brest J, Greiner S, Boskovic B, Mernik M, Zumer V. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. IEEE Transactions on Evolutionary Computation, 2006, 10(6): 646–657
27. Rahnamayan S, Tizhoosh H R, Salama M M A. Opposition-based differential evolution. IEEE Transactions on Evolutionary Computation, 2008, 12(1): 64–79
28. Noman N, Iba H. Accelerating differential evolution using an adaptive local search. IEEE Transactions on Evolutionary Computation, 2008, 12(1): 107–125
29. Yang Z, Tang K, Yao X. Large scale evolutionary optimization using cooperative coevolution. Information Sciences, 2008, 78(15): 2985–2999