Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 51-18 (2018) 269–274

A Hybrid PSO Based on Dynamic Clustering for Global Optimization

Li Hongru, Hu Jinxing, Jiang Shouyong

College of Information Science and Engineering, Northeastern University, Shenyang, Liaoning, 110819, PR China (e-mail: lihongru@ise.neu.edu.cn).

Abstract: Particle swarm optimization is a population-based global search method, and known to suffer from premature convergence prior to discovering the true global minimizer for global optimization problems. Taking balance of local intensive exploitation and global exploration into account, a novel algorithm is presented in the paper, called dynamic clustering hybrid particle swarm optimization (DC-HPSO). In the method, particles are constantly and dynamically clustered into several groups (sub-swarms) corresponding to promising sub-regions in terms of similarity of their generalized particles. In each group, a dominant particle is chosen to take responsibility for local intensive exploitation, while the rest are responsible for exploration by maintaining diversity of the swarm. The simultaneous perturbation stochastic approximation (SPSA) is introduced into our work in order to guarantee the implementation of exploitation and the standard PSO is modified for exploration. The experimental results show the efficiency of the proposed algorithm in comparison with several other peer algorithms.

Keywords: Dynamic clustering; Modified PSO; Exploitation and exploration; Dominant particle; Generalized particle; Simultaneous perturbation stochastic approximation (SPSA)

1. INTRODUCTION

The particle swarm optimizer (PSO), which is one of the population-based algorithms and inspired by the social behavior of animals such as fish schooling and bird flocking in nature, was invented by Kennedy and Eberhart (1995). The original PSO model is very simple, and just utilizes velocity and position as the crucial information of the particle. Furthermore, compared with genetic algorithm (GA), simulated annealing (SA), ant colony optimization (ACO) and some other intelligent algorithms, PSO has the advantage of fewer parameters to be adjusted, better robustness, faster convergence and so on. Therefore, PSO has been applied successfully to several optimization problems, such as global optimization problems, scheduling, and other applications in engineering.

In the context of evolutionary computation, the performance of any global optimization algorithms heavily depends on the mechanism of balancing the two conflicting objectives, which are exploiting the best solutions found so far and at the same time exploring the search space for promising solutions. To reach this balance, many modified PSO algorithms have been proposed in recent years. Nevertheless, utilizing PSO to effectively solve the problem of multiple local minimum, especially for the multi-modal and multi-dimensional optimization, is still a challenge for researchers. Therefore, many researchers have subsequently proposed a lot of different improved strategies which can be divided into the following four categories: initialization (Richards and Ventura,2004), parameter setting (Liu et al.,2016), neighborhood topology (Wang et al.,2013), and hybrid strategy (Moradi and Gholampour,2016). Kennedy (2000) proposed a PSO that uses a K-means clustering algorithm to identify the centers of different clusters of particles in the population. Li and Yang (2009) proposed a clustering PSO (CPSO) by using a hierarchical clustering method to locate and track multiple peaks in dynamic environments. They also proposed a simplified version of CPSO in (Li and Yang, 2010). However, the above clustering methods used to generate sub-swarms only employ the position information of swarm, and neglected the probability that the fitness of two particles may be significantly different while they are close to each other in space, thus these two particles should belong to different levels of sub-swarms. Furthermore, there is no need using all particles within promising sub-regions to perform exploitation to track the potential local minimum, only a representative particle is enough, and the rest take responsibility for exploration.

This paper introduces a novel dynamic clustering method called Dynamic Clustering HPSO (DC-HPSO) algorithm by taking advantage of position information as well fitness of particles. Through clustering, the entire swarm is divided into several sub-swarms. Simultaneous perturbation stochastic approximation (SPSA) (Spall,1987) as a simple yet powerful search technique is used to drive the dominant particles to approach to the potential local optimum, while the standard PSO is remedied to help non-dominant particles to fly away from the dominant particle and even out of its cluster so that much more potential local minimum can be found. At the end of given times of clustering, global optimum can be achieved from the set of potential local minima.

The remaining sections of this paper are arranged as following. Section 2 describes the proposed DC-HPSO algorithm. And experimental results are evaluated on standard test functions in comparison with some peer algorithms taken from the

2405-8963
10.1016/j.ifacol.2018.09.311

literature in Section 3. Finally, Section 4 summarizes the concluding remarks and future work of this study.

2. PROPOSED ALGORITHM

In the traditional PSO algorithm, each particle has a position and a velocity. P_{best} and G_{best} are the best solution for each particle and the global best solution founded by all particles so far, respectively. However, G_{best} does not always guide other particles towards better places if G_{best} is a false global optimum. This phenomenon becomes much more obvious at the last stage of algorithm because of loss of swarm diversity. Besides, a large amount of exploration may be a time-consuming task as well as increase of the complexity of algorithm. How to efficiently get all the potential local minimums is troublesome matter of exploration. Therefore, we proposed DC-HPSO algorithm to solve this problem.

2.1 Idea of Dynamic Clustering in PSO

The introduction of clustering algorithm into PSO can classify the N particles into N_d clusters. Each cluster is composed of particles with similar property and patterns. The information of different clusters is different, but different information of the particles should be fully utilized to promote evolution of the entire swarm. It is this idea that inspires us to use clustering algorithm to enhance the evolution ability of the entire swarm.

Before introduction of clustering, we should explain some definitions here. Dominant particle, denoted by $y_k (k = 1, 2, \cdots, N_d)$, has the following features: first, dominant particle should have fitness as good as possible; second, dominant particle can find itself a better position and help other particles in its neighborhood to evolve at next iteration; third, dominant particle has the superiority of evolution. Generalized particle, which is the particle x_i with its fitness f_i, and denoted by z_i, and $z_i=\left(x_i^T, f_i\right)^T$. This is to say, z_i is a $(n+1)-dimensional$ vector by adding the fitness. Using z_i instead of x_i as unit pattern for clustering has the following advantage: x_i only represents the spatial information of a particle, when particles with little differences of spatial distance while much difference of function values are clustered into the same cluster, multiple local minima may exist, the dominant particle may not find the best local minima, and exploitation of the dominant particle terminates because local minima of sub-region happens, thus the idea of clustering cannot work in this situation. While z_i can avoid the above problem effectively because it is composed of both spatial information and fitness. Here, z_i is needed to be normalized when clustering in order to avoid a bias.

Now, let $Z=\{z_1,z_2,\cdots,z_N\}$ be a set of N generalized particles, each having n features. A partition clustering algorithm tries to find a partition $C=\{C_1,C_2,\cdots,C_{N_d}\}$ of N_d classes. Since the given set can be partitioned by many ways. The most popular way to evaluate similarity between two patterns will be the use of distance measure. The most widely used measurement is the Euclidean distance, which between any two $(n+1)-dimensional$ patterns z_i and z_j is given by,

$$d(z_i, z_j)=\sqrt{\sum_{r=1}^{n+1}\left(z_{i,r}-z_{j,r}\right)^2}=\left\|z_i-z_j\right\| \tag{1}$$

where $\mathrm{d}(z_i, z_j)$ denotes the distance between z_i and z_j.In the following, we will give a brief description of the basic K-means algorithm. First of all, the initial centers should be given to N_d clusters. Then the samples $\{z_i\}$should be distributed to the clusters. The distribution can be conducted by the relation, for all $j=1,2,\cdots,N_d$,

$$z_i \in C_k(t) \text{ if } d(z_i, c_k) < d(z_i, c_j) \tag{2}$$

The quality of the clustering is determined by the following cost function:

$$E=\sum_{j=1}^{N_d}\sum_{z_i\in C_j}\left|z_i-c_j\right|^2 \tag{3}$$

The new cluster centers $c_k(t+1)$, $k=1,2,\cdots,N_d$, at the $(t+1)$-th iterative step, should be computed such that the sum of the squared distances from all points in $C_k(t)$ to the new cluster center is minimized. The measurement which minimizes this is simply the sample mean of $C_k(t)$. Therefore, the new cluster center is given by,

$$c_k(t+1)=\frac{1}{N_k}\sum_{z\in C_{k(t)}} z \ , \quad k=1,2,\cdots,K \tag{4}$$

where N_k is the number of samples in $C_k(t)$. A high level description of the basic K-means is presented in Algorithm 1.

Algorithm 1 K-means Clustering

1: Initialize N_d cluster centers: $c_1,c_2,\cdots,c_{N_d}$, from the normalized set Z ;
2: Set $t=0$, and compute E(t) by using (3);
3: do
4: Set $t=t+1$;
5: for ($i = 1$ *to* N) do
6: Assign z_i to an appropriate cluster by using (2);
7: End for
8: for ($k=1$ *to* N_d) do
9: Update current C_k by using (4);
10: End for
11: Compute the cost function $E(t+1)$ by using (3);
12: while ($E(t)!=E(t+1)$)
13: End while.

By clustering the generalized particles, particles with closeness of spatial location and little difference of fitness can be probably clustered into the same cluster, and local neighborhood is obtained. We sort the fitness of all particles in the cluster, and choose the current best particle as the dominant particle. Then dominant particle is closest to local minimum and can be seen as a representative of the cluster. It's a wise choice to use the only dominant particle to move towards the local minimum of the cluster, while other non-dominant particles should try to fly away from the dominant particle, even jump out of this cluster to explore better positions. Based on this analysis, two methods are introduced into exploitation and exploration respectively: one is SPSA because of its simplicity and fast convergence, the other is a modified PSO.

2.2 SPSA Technique for Dominant Particle

One optimization method that has attracted considerable international attention is simultaneous perturbation stochastic approximation (SPSA) method. As motivated by not using direct measurements of the gradient of the objective function

which are often difficult or impossible to obtain, SPSA uses only objective function measurements, which is quite different from methods such as SA or GA. Further, SPSA is especially efficient in high-dimensional problems in terms of providing a good solution at the cost of a relatively small number of measurements of the objective function (Spall,1987). The SPSA procedure is in the general recursive stochastic approximation form:

$$x_{k+1}=x_k - a_k\bar{g}_k(x_k) \tag{5}$$

where $\bar{g}_k(x_k)$is the estimate of the gradient $g(x_k)$ and a_k is the gain sequence satisfying certain conditions at the *kth* iteration. $\bar{g}_k(x_k)$ for a two-sided finite-difference approximation is given by,

$$\bar{g}(x_k)=\frac{f(y_k+c_k\Delta_k)-f(y_k-c_k\Delta_k)}{2c_k}\begin{bmatrix}\Delta_{k1}^{-1}\\ \Delta_{k2}^{-1}\\ \vdots\\ \Delta_{kn}^{-1}\end{bmatrix} \tag{6}$$

where Δ_k denotes a vector independently generated from a zero-mean probability distribution, Δ_{ki} is the *ith* component of Δ_k, and c_k denotes a small positive number that usually get smaller as k gets larger. The basic SPSA algorithm we implemented is presented in Algorithm 2.

Algorithm 2 SPSA

1: Initialization and coefficient selection for x_1, *IterNo*, a, c, A, α, γ ;
2: for $\forall\ k \in [1, IterNo]$ do
3: Generate zero-mean, *n-dimensional* perturbation vector Δ_k;
4: Generate two small constants: $a_k = a/(A+k)^{\alpha}$ and $c_k = c/k^{\gamma}$;
5: Compute $f(x_k+c_k\Delta_k)$ and $f(x_k - c_k\Delta_k)$;
6: Compute pseudo-gradient $\bar{g}_k(x_k)$ by using (6);
7: Compute x_{k+1} using (5);
8: End For.

2.3 Modified PSO for Non-dominant Particle

As analyzed earlier, in a fixed cluster, says C_k , where y_k is the dominant particle, the non-dominant particle should fly away from the dominant particle as far as possible and even out of the real space of this cluster. Thus, non-dominant particle x_i at time t would possibly fly along the opposite direction of $(y_k - x_i^t)$. Sketch map of iteration for non-dominant particle in its cluster is presented in Fig.1.

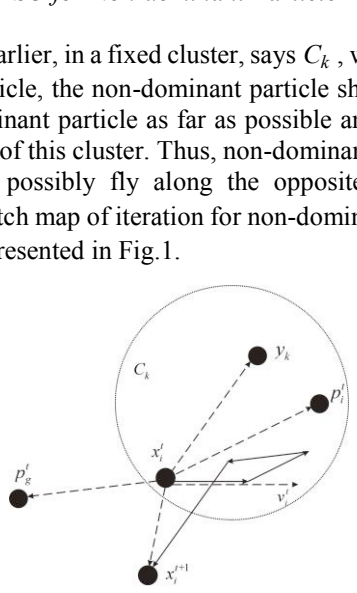

Fig. 1. Sketch map of iteration for non-dominant particle.

Taking the original velocity of x_i into comprehensive consideration, x_i can be updated as follows:

$$v_{i,j}^{t+1}=wv_{i,j}^t + c_1r_1\left(p_{i,j}^t - x_{i,j}^t\right)+c_2r_2\left(p_g^t - x_{i,j}^t\right)+s_k\left(x_{i,j}^t - y_k\right) \tag{7}$$

$$x_{i,j}^{t+1}=x_{i,j}^t+v_{i,j}^{t+1} \tag{8}$$

where $x_{i,j}^t$ and $v_{i,j}^t$ are the *jth* dimensional component of the position and velocity of particle i at time t, respectively; $p_{i,j}^t$ and $p_{g,j}^t$ are the *jth* dimensional component of P_{best} of particle i and G_{best} at time t, respectively; c_1 and c_2 are positive accelerating constants. r_1 and r_2 are two random variables with a uniform distribution in the range of [0,1]. w is the inertia weight. s_k is the gain sequence, which is a positive number that can control the diversity of swarm. A large one can enhance the diversity of swarm, while a small one leads to fast convergence. User can freely choose s_k from any probability distribution. The modified PSO algorithm for non-dominant particles is described in Algorithm 3.

Algorithm 3 Modified PSO

1: Initialize parameters of N, *ToIterNo*, V_{max}, s_k;
2: Calculate the fitness and generate P_{best} and G_{best};
3: for (t=1 *to ToIterNo*) do
4: for (i=1 *to N*) do
5: for (j=1 *to n*) do
6: Compute $v_{i,j}^{t+1}$ by using (7);
7: if $\left(v_{i,j}^{t+1} > V_{max}\right)$ then do
8: $v_{i,j}^{t+1} = V_{max}$;
9: else if $\left(v_{i,j}^{t+1} < -V_{max}\right)$ then do
10: $v_{i,j}^{t+1} = -V_{max}$;
11: End if
12: Compute $x_{i,j}^{t+1}$by using (8);
13: End for
14: Evaluate the fitness $f\left(x_i^{t+1}\right)$;
15: Update P_{best};
16: End for
17: Update G_{best};
18: End for.
19: Output G_{best} .

2.4 Framework of Dynamic Clustering HPSO Algorithm

There are two features making the proposed algorithm dynamic. First, after each time of clustering, particles in each cluster as a sub-swarm exploit or explore according to their roles. Thus, SPSA and modified PSO algorithms need to be performed a number of iterations in each cluster until both satisfy the termination conditions. The next behavior of clustering takes place until all the clusters finish the exploitation and exploration of the current period of clustering. Second, a memory array should be established to store the positions and fitness of the N_d old dominant particles. At next time of clustering, an old dominant particle determines which newly produced cluster it should belong to. If the new dominant particle is superior to the old one, the old is replaced. If the swarm converges, the N_d particles will be stored in the same cluster. Therefore, we call it DC-HPSO algorithm. According to the analysis mentioned above, the complete description of DC-HPSO is presented in Algorithm 4.

Algorithm 4 DC-HPSO

1: Initialize parameters *CluIterNo*, N_d, and initialize particle swarm;
2: Choose N_d particles and memory them into the array of $M^{(0)}$;
3: for (t=1 *to CluIterNo*) do
4: Obtain N_d clusters from set $\{z_i\}$ by using Algorithm 1;
5: for (k=1 *to* N_d) do
6: Choose the dominant particle y_k from cluster C_k;
7: Execute Algorithm 2 for dominant particle y_k;
8: Execute Algorithm 3 for non-dominant particles;
9: Update dominant particle y_k as well as $M_k^{(t)}$;
10: if ($M_k^{(t-1)}$ is superior to $M_k^{(t)}$) then do
11: $M_k^{(t)} = M_k^{(t-1)}$;
12: End if
13: End for
14: Update G_{best} according to $M^{(t)}$;
15: End for.

3. EXPERIMENTS

3.1 Test Functions and Algorithms Compared

The proposed algorithm was applied to the 8 well-known boundary constrained benchmarks (Sun et al., 2004) and other four functions (Bergh, 2002) to evaluate the performance. All test functions were presented in Table 1. The 12 test functions are divided into three groups in terms of their properties: uni-modal and multi-dimensional problem$(f_1 - f_4)$, multi-modal and multi-dimensional problems $(f_5 - f_8)$, traditional multi-modal and low-dimensional problems $(f_9 - f_{12})$. For function f_{11}, $a(i)$=16(i *mod* 5 − 2), $b(i)$=16([i/5] − 2).

Simulations were carried out to achieve a comparative performance analysis of the proposed DC-HPSO algorithm with respect to:

(i). the standard PSO (SPSO) (Richards and Ventura,2004)
(ii). ARPSO (Riget and Vesterstrøm, 2002)
(iii). Quantum-behaved PSO (QPSO) (Sun et al., 2004)
(iv). Multi-start PSO (MPSO) (Bergh, 2002)
(v). GCPSO (Bergh, 2002)
(vi). RePSO (Evers and Ben, 2009).

3.2 Numerical Results and Comparisons

The comparative study presented on the 12 test functions, focuses on the following performance metrics: (a) the quality of the final solution; (b) the frequency of striking the optima; (c) the convergence of HPSO with different parameter settings. In all algorithms, for a particular trial, the same initial positions and velocities were set for all particles, so as to minimize the effect of randomness during comparison. In the simulation, all algorithms used the global version of PSO. Statistics results from 100 trials per benchmark per algorithm over 60,000 function evaluations per trial using swarm size 40, ω=0.72894, c_1=c_2=1.49618 , which were obtained using Clerc's constriction models (Clerc and Kennedy, 2002), as for DC-HPSO, *ToIterNo*=50 in Algorithm 3, A=1 , a=1 , α=0.602 , r=0.101 according to literature (Spall,2005) and *IterNo*=50 in Algorithm 2, *ClusterNo* =30 in Algorithm 4.

The relative statistics results of DC-HPSO where s_k was set as 0.5, and the number of clusters, N_d was set as 5. Comparisons in terms of quality of the final solutions among functions $f_1 - f_{12}$were given in Table 2, where "Mean" indicated the mean best solutions found in the last generation and "Std. Dev" denoted the standard deviation. "Best" and "Worst" were the best and worst fitness value throughout 100 trials, respectively. From the results, DC-HPSO outperforms all the other peer PSO algorithms on functions $f_1 - f_{10}$. While for functions $f_{11} - f_{12}$, GCPSO and MPSO outperform the DC-HPSO algorithm by a narrow margin. This means DC-HPSO improves the quality of the average optima in most cases.

Table 1 Numerical test functions

Test function	D	Range	f_{min}
f_1=$\sum_{i=1}^{D}(x_i - 1.24)^2$	30	$[-100,100]^D$	0
f_2=$\sum_{i=1}^{D-1}\left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right]$	30	$[-30,30]^D$	0
f_3=$\sum_{i=1}^{D}\lvert x_i\rvert + \prod_{i=1}^{D}\lvert x_i\rvert$	30	$[-10,10]^D$	0
f_4=$-20exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right) - exp\left(\frac{1}{D}\sum_{i=1}^{D}cos(2\pi x_i)\right) + 20 + e$	30	$[-32,32]^D$	0
f_5=$\sum_{i=1}^{D}\left[x_i^2 - 10cos(2\pi x_i) + 10\right]$	30	$[-5.12,5.12]^D$	0
f_6=$\frac{1}{4000}\sum_{i=1}^{D}x_i^2 - \prod_{i=1}^{D}cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	$[-600,600]^D$	0
f_7=$\sum_{i=1}^{D}\left(\sum_{j=1}^{i}x_j\right)^2$	30	$[-100,100]^D$	0
f_8=$\frac{1}{D}\sum_{i=1}^{D}x_i sin\sqrt{\lvert x_i\rvert} + 418.983$	30	$[-500,500]^D$	0
f_9=$1 - exp\left(-2log(2)\times\left(\frac{x-0.08}{0.854}\right)\right)\times sin^6\left(5\pi\left(x^{3/4} - 0.05\right)\right)$	1	$[0,1]$	0
f_{10}=$0.5 + \frac{sin^2\left(\sqrt{x^2+y^2}\right) - 0.5}{1 + 0.001(x^2+y^2)}$	2	$[-10,10]^D$	0
f_{11}=$500 - \frac{1}{0.002 + \sum_{i=0}^{24}\frac{1}{1+(x-a(i))^6 + (y-b(i))^6}}$	2	$[-65.535,65.535]^D$	0
f_{12}=$\left\{\sum_{i=1}^{5}icos[(i+1)x+i]\right\}\cdot\left\{\sum_{i=1}^{5}icos[(i+1)y+i]\right\} + (x+1.42513)^2 + (y+0.80032)^2 + 186.7$	2	$[-10,10]^D$	0

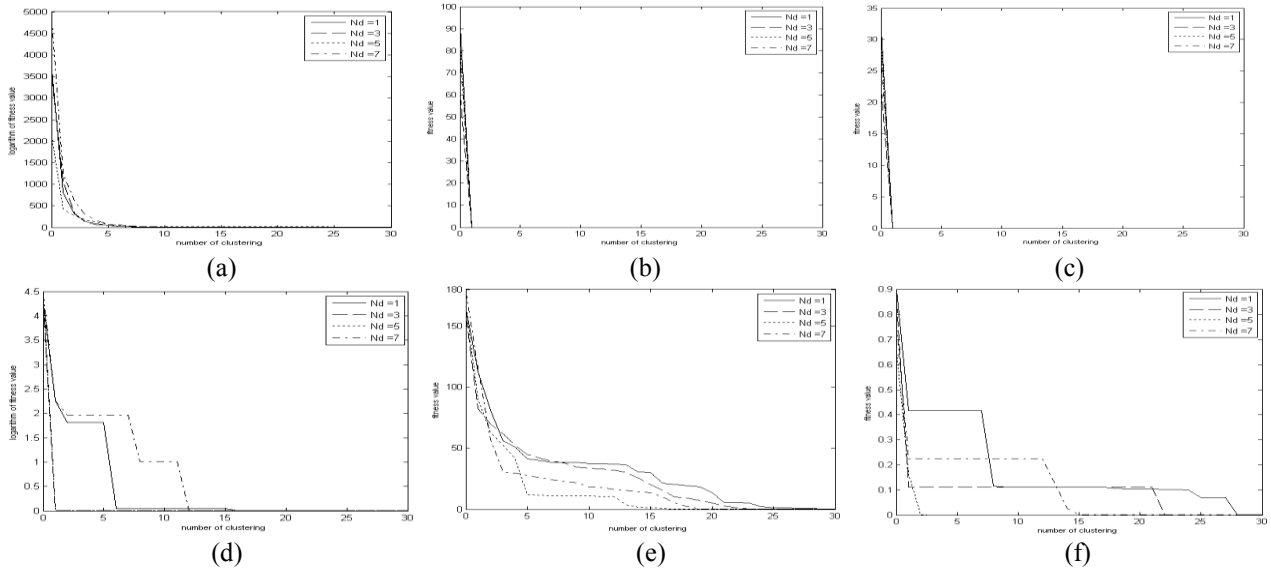

Fig. 2. Performance of DC-HPSO for functions $f_1 - f_6$ using different values of N_d : (a) the performance of DC-HPSO for f_1, (b) the performance of DC-HPSO for f_2 ,(c) the performance of DC-HPSO for f_3 , (d) the performance of DC-HPSO for f_4, (e) the performance of DC-HPSO for f_5 , (f) the performance of DC-HPSO for f_6 .

Table 2 Comparison of algorithms for test functions

Function		SPSO	GCPSO	MPSO	QPSO	ARPSO	RePSO	**DC-HPSO**
f_1	Mean	5.89	1.75e-027	3.95e-024	1.35e-022	43.201	1.75e-004	**4.44e-032**
	Std. Dev	1.815	1.32e-027	1.84e-021	4.52e-022	2.84e-014	0	**3.45e-032**
	Best	2.155	7.67e-028	7.73e-024	1.18e-029	43.201	1.75e-004	**0**
	Worst	9.745	4.93e-027	2.55e-018	2.17e-021	43.201	1.75e-004	**9.86e-032**
f_2	Mean	61.625	21.380	9.242	36.359	29	27.353	**0.118**
	Std. Dev	13.522	2.442	1.799	32.500	0	7.430	**0.321**
	Best	35.301	15.354	2.337	20.042	29	19.885	**9.27e-007**
	Worst	96.976	27.528	12.683	180.119	29	32.349	**1.761**
f_3	Mean	0.380	2.30e-004	3.49e-007	7.21e-024	0	5.085	**0**
	Std. Dev	0.240	2.25e-002	1.24e-006	2.38e-023	0	0.236	**0**
	Best	0.043	1.26e-004	9.87e-008	2.18e-028	0	6.212	**0**
	Worst	1.060	5.93e-004	7.47e-007	1.10e-022	0	4.554	**0**
f_4	Mean	1.122	1.778	2.408	3.22e-012	8.88e-016	0.101	**8.88e-016**
	Std. Dev	0.342	0	0.992	5.89e-012	0	0.745	**0**
	Best	0.597	1.778	4.587	1.47e-013	8.88e-016	0.073	**8.88e-016**
	Worst	1.881	1.778	3.488	2.89e-011	8.88e-016	1.324	**8.88e-016**
f_5	Mean	66.659	46.763	52.733	20.118	0	53.452	**0**
	Std. Dev	13.316	4.877	3.466	5.196	0	0.846	**0**
	Best	36.586	12.656	17.748	10.964	0	51.738	**0**
	Worst	87.967	53.768	65.546	32.837	0	55.154	**0**
f_6	Mean	0.095	0	0.007	0.014	2	0.015	**0**
	Std. Dev	0.042	0	0.712	0.013	0	0.047	**0**
	Best	0.038	0	3.75e-003	0.002	2	4.49e-004	**0**
	Worst	0.228	0	0.355	0.051	2	0.825	**0**
f_7	Mean	6.57e-031	1.414	0.485	0.234	0	1.83e+003	**0**
	Std. Dev	3.54e-030	0.023	0.354	0.673	0	26.458	**0**
	Best	0	0.572	0.271	0.082	0	1.57e+003	**0**
	Worst	1.97e-029	2.019	0.759	1.343	0	2.43e+003	**0**
f_8	Mean	3.87e+002	36.465	138.715	1.21e+002	2.052	87.244	**5.70e-004**
	Std. Dev	59.535	5.424	11.411	38.052	0	32.451	**1.94e-003**
	Best	2.27e+002	9.536	100.423	71.314	2.052	12.932	**1.29e-009**
	Worst	4.16e+002	145.197	256.764	2.18e+002	2.052	130.178	**0.01076**
f_9	Mean	2.96e-004	3.73e-003	0.899	9.34e-004	0.029	1.271	**1.97e-015**
	Std. Dev	0	0.056	0.464	0	0	0.438	**0**
	Best	2.96e-004	1.44e-004	0.052	9.34e-004	0.029	0.899	**1.97e-015**
	Worst	2.96e-004	0.133	1.774	9.34e-004	0.029	2.467	**1.97e-015**
f_{10}	Mean	4.69e-004	5.80e-128	4.04e-123	5.31e-004	0	4.45e-008	**0**
	Std. Dev	7.17e-004	0	0	7.32e-004	0	5.84e-006	**0**
	Best	0	5.80e-128	4.04e-123	8.48e-008	0	1.23e-014	**0**
	Worst	1.57e-003	5.80e-128	4.04e-123	1.56e-003	0	5.41e-005	**0**

f_{11}	Mean	5.831	0	0	9.85e-004	9.85e-004	0	**9.85e-004**
	Std. Dev	4.436	0	0	0	0	0	**0**
	Best	9.85e-004	0	0	9.85e-004	9.85e-004	0	**9.85e-004**
	Worst	9.207	0	0	9.85e-004	9.85e-004	0	**9.85e-004**
f_{12}	Mean	0.311	7.06e-002	2.13e-003	0.103	1.91e+002	0.779	**0.163**
	Std. Dev	0.382	0.012	3.57e-003	0.265	9.114	0.043	**0.279**
	Best	9.08e-004	9.09e-004	9.09e-004	9.09e-004	1.53e+002	0.067	**9.08e-004**
	Worst	0.780	0.143	3.09e-002	0.780	1.94e+002	0.981	**0.780**

Figure 2 depict the performance of DC-PSO with different number of clusters for 6 selected functions over the evolution process on 100 trial runs, where s_k was set as 0.5. N_d =1 means dynamic clustering is not used for the tests. The experimental results on 6 functions show that our dynamic clustering technique can make the convergence of DC-HPSO faster (especially when N_d was set as 5), as well as achieve better results compared with no use of this technique in most cases. The results showed that in most cases, the proposed algorithm could achieve good performance. The advantage of our proposed algorithm may be owing to its local search ability as well as global search ability, since for the optimization problem, trading off between exploration and exploitation during the search is critical to the performance.

4. CONCLUSIONS

This paper presented a novel dynamic clustering HPSO algorithm which firstly clusters the similar particles into the same sub-region and then uses the SPSA and modified PSO algorithms to perform the jobs of exploitation and exploration, respectively. Here, we have achieved this goal by defining dominant particle which can take responsibility for exploitation so that solutions can be refined. Together with the non-dominant particles, which are responsible for exploration, thus the diversity can be maintained effectively. Our approach shows a good performance and outperforms several peer PSO algorithms for most of the studied problems. Hence, we can conclude that our proposed algorithm could achieve a suitable balance between enhancing population diversity and refining solutions. Our experiments are based on specific functions. In the future, experiments in real-world applications will be indispensable for verifying the efficiency of our algorithm.

ACKNOWLEDGMENTS

This work was supported by the National Nature Science Foundation of China [Grant number 61533007].

REFERENCES

J. Kennedy and R. Eberhart. (1995). Particle swarm optimization. *Proceeding IEEE International Conference on Neural Networks,* 4, pp.1942–1948.

M. Richards and D. Ventura. (2004). Choosing a starting configuration for particle swarm optimization. *2004 IEEE International Joint Conference on Neural Networks,* 3, pp.2309–2312.

J. Liu, Y. Mei and X. Li. (2016). An analysis of the inertia weight parameter for binary particle swarm optimization. *IEEE Transactions on Evolutionary Computation,* 20, pp. 666–681.

H. Wang, H. Sun and C. Li. (2013). Diversity enhanced particle swarm optimization with neighborhood search. *Information Sciences,* 223, pp. 119–135.

P. Moradi and M. Gholampour. (2016). A hybrid particle swarm optimization for feature subset selection by integrating a novel local search strategy. *Applied Soft Computing,* 43, pp. 117–130.

J. Kennedy. (2000). Stereotyping: improving particle swarm performance with cluster analysis. *Proceedings of the 2000 Congress on Evolutionary Computation,* 2, pp.1507–1512.

C. H. Li and S. X. Yang. (2009). A clustering particle swarm optimizer for dynamic optimization. *IEEE Congress on Evolutionary Computation,* pp.439–446.

S. X. Yang and C. H. Li. (2010). A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments. *IEEE Transactions on Evolutionary Computation,* 14, pp. 959–974.

Spall J.C. (1987). A stochastic approximation technique for generating maximum likelihood parameter estimates. *American Control Conference,* pp.1161–1167.

Blum C. and Roli A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys,* 35, pp.268–308.

J. Riget and J. S. Vesterstrøm. (2002). A diversity-guided particle swarm optimizer-the ARPSO. *EVALife Technical Report,* pp. 1570–1575.

J. Sun, B. Feng and W.B. Xu. (2004). Particle swarm optimization with particles having quantum behaviour. *IEEE Congress on Evolutionary Computation,* 1, pp. 325–331.

Van den Bergh, F. (2002). An analysis of particle swarm optimizers. Ph.D. Thesis, Department of Computer Science, University of Pretoria, Pretoria, South Africa.

Van den Bergh, F and Engelbrecht A P. (2002). A new locally convergent particle swarm optimise. *IEEE Conference on Systems, Man and Cybernetics,* 3, pp. 96–101.

Evers G.I. and Ben Ghalia M. (2009). Regrouping particle swarm optimization: a new global optimization algorithm with improved performance consistency across benchmarks. *IEEE International Conference on Systems, Man and Cybernetics,* pp.3901–3908.

M. Clerc and J. Kennedy. (2002). The particle swarm-explosion, stability, and convergence in multidimensional complex space. *IEEE Transactions on Evolutionary Computation,* 6, pp. 58–73.

Spall J.C. (2005). Adaptive stochastic approximation by the simultaneous perturbation method. *IEEE Transactions on Automatic Control,* 45, pp. 1839–1853.