A Clustering Particle Swarm Optimizer for Dynamic Optimization

Changhe Li and Shengxiang Yang

***Abstract*—In the real world, many applications are non-stationary optimization problems. This requires that optimization algorithms need to not only find the global optimal solution but also track the trajectory of the changing global best solution in a dynamic environment. To achieve this, this paper proposes a clustering particle swarm optimizer (CPSO) for dynamic optimization problems. The algorithm employs hierarchical clustering method to track multiple peaks based on a nearest neighbor search strategy. A fast local search method is also proposed to find the near optimal solutions in a local promising region in the search space. Six test problems generated from a generalized dynamic benchmark generator (GDBG) are used to test the performance of the proposed algorithm. The numerical experimental results show the efficiency of the proposed algorithm for locating and tracking multiple optima in dynamic environments.**

I. Introduction

Most research in evolutionary computation focuses on static optimization problems. However, many real-world problems are dynamic optimization problems (DOPs), where changes occur over time. This requires optimization algorithms to not only find the optimal solution in a short time but also track the optimal solution in a dynamic environment. Hence, optimization methods that are capable of continuously adapting the solution to a changing environment are needed. Particle swarm optimization (PSO) is a versatile population-based stochastic optimization technique. Similar to other evolutionary algorithms (EAs) in many respects, PSO has been shown to perform well for many static problems [17], and several PSO based algorithms have been recently proposed to address DOPs [3], [15], [21].

It is difficult for the standard PSO to optimize DOPs. The difficulties lie in two aspects: one is the outdated memory due to the changing environment and the other is the diversity loss due to convergence. Of these two difficulties, the diversity loss is by far more serious [3]. It has been demonstrated that the time taken for a partially converged swarm to re-diversify, find the shifted peak, and then re-converge is quite deleterious to the performance of PSO [1].

In standard PSO, the diversity loss is due to the attraction of global best particle which attracts all the particles quickly converging on local optima. However, the aim of dynamic optimization is not only to track global optima but also to find local optima. In order to detect local optima as many as possible, it is important that how to guide particles searching in different promising regions. The promising local optima are those local optima that any one of them probably becomes the global best in next new environment, so local best-fit particles are needed to guide the search in local regions of the search space. However, the question is how to determine which particles would be suitable as the neighborhood best and how to assign particles in different neighborhood to move toward different local region.

This paper proposes a clustering PSO (CPSO) that can adaptively detect local regions and assign particles in different neighborhoods. In CPSO, there are two search strategies: global search and local search. For global search, the aim is to cover promising local optima as many as possible. To achieve it, we change the learning mechanism in standard PSO that each particle learns information from its own history best position and the history best position of its nearest neighbor other than the global best one. This strategy enables particles to detect the local search region around themselves and assign them in different neighborhoods. By using clustering method, the whole swarm can be adaptively divided into a number of sub-swarms which cover different local regions. The clustering method we used is single linkage hierarchical clustering method. In order to accelerate local search, a new learning strategy for the global best particle is introduced in CPSO.

The rest of the paper is organized as follows: Section II introduces the particle swarm optimization and some research of multi-swarm techniques in dynamic environments. The CPSO is present in section III. Section IV presents the experimental study and discussions. Finally, conclusions are given in section V.

II. Relevant Work

A. Particle Swarm Optimization

Particle Swarm Optimization (PSO) was first introduced by Kennedy and Eberhart in [9], [10]. PSO is motivated from the social behavior of organisms, such as bird flocking and fish schooling. In PSO, a swarm of particles "fly" through the search space. Each particle follows the previous best position found by its neighbor particles and the previous best position found by itself. In the past decade, PSO has been actively studied and applied for many academic and real world problems with promising results due to its property of fast convergence [18].

Ever since PSO was first introduced, several major versions of the PSO algorithms have been developed [18]. Each particle is represented by a position and a velocity, which are updated as follows:

$$V'^{d}_{i} = \omega V^{d}_{i} + \eta_1 r_1 (pbest^{d}_{i} - X^{d}_{i}) + \eta_2 r_2 (gbest^{d} - X^{d}_{i}) \quad (1)$$

$$X'^{d}_{i} = X^{d}_{i} + V'^{d}_{i}, \quad (2)$$

The authors are with the Department of Computer Science, University of Leicester, University Road, Leicester LE1 7RH, United Kingdom (email: {cl160, s.yang}@mcs.le.ac.uk)

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) of the United Kingdom under Grant EP/E060722/1.

where ${X'}_i^d$ and X_i^d represent the current and previous position of $d-th$ dimension of particle i respectively, ${V'}_i$ and V_i are the current and previous velocity of particle i respectively, $pbest_i$ and $gbest$ are the best position found by particle i so far and the best position found by the whole swarm so far respectively, $\omega \in (0, 1)$ is an inertia weight, which determines how much the previous velocity is preserved, η_1 and η_2 are the acceleration constants, and r_1 and r_2 are random numbers generated in the interval $[0.0, 1.0]$.

From the theoretical analysis of the trajectory of a PSO particle [8], the trajectory of a particle $\vec{X}_i$ converges to a weighted mean of $\vec{P}_i$ and $\vec{P}_g$. Whenever the particle converges, it will "fly" to the individual best position and the global best position. According to the update equation, the individual best position of the particle will gradually move closer to the global best position. Therefore, all the particles will converge onto the global best particle's position.

There are two main models of the PSO algorithms, called $gbest$ (global best) and $lbest$ (local best), which differs in the way of defining the neighborhood of each particle. In the $gbest$ model, the neighborhood of a particle consists of the particles in the whole swarm, which share information between each other. On the contrary, in the $lbest$ model, the neighborhood of a particle is defined by several fixed particles. The two models give different optimization performances on different problems. Kennedy and Eberhart [12] and Poli et al. [18] pointed out that the $gbest$ model has a faster convergence speed with a higher chance of getting stuck in local optima than $lbest$. On the contrary, the $lbest$ model is less vulnerable to the attraction of local optima but with a slower convergence speed than the $gbest$ model.

B. Multiple Swarms

Many researchers have considered multi-populations as a means of enhancing the diversity of EAs to address DOPs. Branke et al. proposed a self organizing scouts (SOS) [5] algorithm that has been shown to give excellent results on the many peaks benchmark. Parrott and Li developed a speciation based PSO (SPSO) [16], which dynamically adjusts the number and size of swarms by constructing an ordered list of particles, ranked according to their fitness, with spatially close particles joining a particular species.

The atomic swarm approach has been adapted to track multiple optima simultaneously with multiple swarms in dynamic environments by Blackwell and Branke [2], [3]. In their approach, a charged swarm is used for maintaining the diversity of the swarm, and an exclusion principle ensures that no more than one swarms surround a single peak. This strategy is very efficient for the moving peaks benchmark (MPB) function [4].

Kennedy [11] proposed a PSO that uses a k-means clustering algorithm to identify the centers of different clusters of particles in the population, and then use these cluster centers to substitute the personal bests or neighborhood bests. Brits et al. [6] proposed a *nbest* PSO algorithm which defines the "neighborhood" of a particle as its closest particles in the population. The neighborhood best for each particle is defined as the average of the positions of its closest particles. In [7], a niching PSO (NichePSO) was proposed by incorporating a cognitive only PSO model and the guaranteed convergence PSO (GCPSO) [19] algorithm.

III. Clustering Particle Swarm Optimization

The traditional method of using the multi-population method to track optima for multi-modal functions is to divide the whole search space into local subspaces which might be covered by one or a small number of local optima, then separately search on these subspaces. However, the difficulty is how to guide particles to move toward different promising sub-regions and how to define the area of each sub-region as well as how many sub-populations are needed. The first question requires that an algorithm should have a good global search capability to explore promising sub-regions. For the second question, if the area of sub-region is too small, there is a potential problem that the small isolated sub-populations may converge on a local optimum. In this case, the diversity will lose and the algorithm can hardly make any progress. However, if a sub-region is too large, there will be more than one peaks within the sub-region of a sub-swarm. It is very hard to know what shape a sub-region is, so particles within the neighborhood should calculate the sub-area by themselves. For the third problem, if too many sub-swarms distribute in the fitness landscape, it may waste the limited computation resources. On the contrast, if there are too small number of sub-swarms, the algorithm can not efficiently track local optima.

To overcome the above problems when using the multi-population method, CPSO employs a global search method to detect promising sub-regions and a hierarchical clustering method to generate a proper number of sub-swarms.

A. Global Search Strategy

The population topology can greatly affect PSO's performance. Most PSO algorithms use a fixed population topology that particles learn information from some fixed neighbors. One fixed topology may be not suitable for all situations, so a dynamic topology is needed. The neighborhood size of a particle is another factor that can affect PSO's performance.

In CPSO, the neighborhood of each particle has only one particle that is the closest to it. Learning from the nearest neighbor enables a particle to explore the region of local optima around itself. Particles that are nearby a local optimum will get closer and closer to that region because the $pbest$ is replaced only when a better position is found. Gradually, they will generate a local cluster around that local optimum. Particles in one local cluster are not influenced by those far away (other local clusters) even they have a very good fitness. This strategy can help swarm find more local optima rather than one optimum that can be usually achieved by the standard PSO, especially for multi-modal problems. The velocity update equation for the global search is as follows:

$$V_i^d = \omega V_i^d + \eta_1 r_i^d (pbest_i^d - X_i^d) + \eta_2 \cdot r_i^d \cdot (pbest_{i\_nearest}^d - X_i^d) \tag{3}$$

where $pbest_{i\_nearest}$ is the $pbest$ of the closest particle to particle i.

Though the neighborhood of a particle has only one particle, this can help PSO find as many promising local regions as possible. If the neighborhood size is greater than one, there may be more than one peaks within the neighborhood. However, all the particles within the neighborhood only learn from the best one and hence, finally they will cover only one peak. So, the tracking information of other peaks is lost. On the other hand, though the neighbor of a particle is the nearest particle to itself, it is not fixed during the search process. The neighbor may change over generation to generation.

B. Hierarchical Clustering Method

Some researchers have used the k-mean clustering method to generate sub-swarms, the problem of the k-mean method is that we do not know the optimum value of k for the current population. Too large or too small k will cause the problem of an improper number of sub-swarms, discussed above. In CPSO, the single-linkage hierarchical clustering method is used to solve this problem.

Before carrying out the clustering operation, the global search method runs for several generations. When the whole population are well distributed in different sub-regions, the clustering operation is carried out. Two clustering methods are proposed in CPSO: rough clustering and refining clustering. For both clustering methods, we define the distance $d(i, j)$ between two particles i and j in the n-dimension space as the Euclidean distance as follows:

$$d(i,j) = \sqrt{\sum_{d=1}^{n}(x_i^d - x_j^d)^2} \tag{4}$$

The radius for each sub-swarm (cluster) is defined as follows:

$$r = \sum_{i=1}^{m} d(i, center)/m \tag{5}$$

where $center$ is the center position of the sub-swarm, m is the population size of the sub-swarm. The distance of two clusters is the distance of two closest particles of the two clusters.

The goal of the rough clustering method is to generate temporary sub-swarms. Each temporary sub-swarm may cover several peaks, so further refining clustering is needed to get more accurate sub-swarms so that each of them covers no more than one peaks. The differences between rough clustering and refining clustering are the stop and merging criteria.

For the rough clustering method, two cluster lists are used to save the clustering results of two successive generations respectively. If two clusters from the two cluster lists respectively are the same (i.e., they have the same particles), then a temporary sub-swarm that is composed of these particles is generated.

For rough clustering, if the following condition is true, then stop the rough clustering operation.

- Stop clutering if the number of particles of all clusters is greater than 1.

Once temporary sub-swarms are produced during the rough clustering method, they are allowed for evolution for num_ref generations using the global search method. Then, the refining clustering operation is carried out on these temporary sub-swarms to obtain the final sub-swarms. For refining clustering, there are three cases for merging two clusters:

Case 1: The number of particles of the two clusters are both greater than 1. In this case, merge them only if the half distance of the two clusters is less than the radius of the cluster with a larger radius.

Case 2: The number of particles of one of the two clusters is equal to 1, the other is greater than 1. In this case, merge them only if the half distance of them is less than the radius of the cluster with particles greater than 1.

Case 3: The number of particles of the two clusters are both equal to 1. If there are no other clusters with more than 1 particles, merge these two clusters. Otherwise, merge the two clusters only if c times the distance between them is less than the radius of the cluster whose radius is the largest in all clusters. Here, the constant c is a random number between $[1.5, 2.0]$.

If the three cases above are not satisfied, then stop clustering.

C. Local Search Strategy

In original PSO, the $gbest$ is updated only when particles find a much better position than the $gbest$, once it is updated, the information of all dimensions of the $gbest$ is replaced with the better one. This updating mechanism has a disadvantage that promising information of some dimensions of one particle couldn't be kept due to bad information in other dimensions that cause its low fitness. This problem is called "Two step forward, one step back" in [20].If a particle gets better, information of some dimension probably becomes more promising. Other particles should learn some useful information from the improved one even the particle's fitness is very low. In CPSO, the $gbest$ learns the useful information from those dimensions of a particle that is improved, once promising information is extract from those improved dimensions of that particle, the information of corresponding dimensions of the $gbest$ is updated. The updating happens only when particles are improved, which in the following method:

The learning method is time comsuming, we can not make all particles using this strategy, so we choose the $gbest$ as the learner for each sub-swarm. Experimental study shows that this strategy makes the convergence speed very fast, which is favorable for dynamic environment.

In local search method, the standard PSO is used with a linear decreasing inertia weight, the framework of local search is described as:

Algorithm 1 GbestUpdate(particle p, fes)

```
1: for each dimension $d$ of $gbest$ do
2:    $\vec{X}_{t\_gbest} := \vec{X}_{gbest}$
3:    $X_{t\_gbest}[d] := X_p[d]$
4:    if $t\_gbest$ is better than $gbest$ then
5:       $X_{gbest}[d] := X_{t\_gbest}[d]$
6:    end if
7:    fes++
8: end for
```

Algorithm 2 LocalSearch(fes)

```
1: for each particle $i$ do
2:    Update the velocity and position according to Eqs. (1)
      and (2)
3:    fes++;
4:    if particle $i$ is better than it's $pbest$ then
5:       Update $pbest_i$
6:       Perform GbestUpdate(i,fes) for $gbest$
7:       if particle $i$ is better than $gbest$ then
8:          Update $gbest$
9:       end if
10:   end if
11: end for
```

D. Framework of Clustering PSO

CPSO starts from an initial swarm named cradle swarm. The global search method is operated on the cradle swarm for num_rgh generations. Then, the rough clustering method is carry out to produce temporary sub-swarms till the cradle swarm is empty. For each temporary sub-swarm, we still use the global search method for num_ref generations. When the number of generations of a temporary sub-swarm is greater than num_ref, the refining clustering operation is carried out to produce final sub-swarms till all temporary sub-swarms finish the refining clustering operation, then local search method is performed on all final temp sub-swarms.

Traditionally, overlapping search between two sub-swarms can be checked by comparing the distance of the best particles of the two sub-swarms, if the distance is less than their search radius, then combine or remove one of them. The above checking mechanism assumes that each sub-swarm just cover one peak. However, it is not true for real PSOs. If any one of those sub-swarms in that single local region covers more than one peaks, those swarms are within their search area shouldn't combine together or remove some of them. In CPSO, if two sub-swarms are within each other search area, a overlapping ratio is calculated, the ratio is the percentages of particles of one sub-swarm which are within another sub-swarm's search radius. The two sub-swarms are combined only both overlapping ratio are greater than a constant value of 0.7.

In dynamic optimization, the best solutions found in the current environment may be useful in the next environment. In CPSO, a converge list $conver\_lst$ is used to record the best particles found in the current environment. If the radius of one sub-swarm is less than a small value $\sigma = 0.0001$, the sub-swarm is regarded to be converged on a local optimum. If a sub-swarm converges, the $gbest$ of it is added into $conver\_lst$. Once an environment change is detected, the particles are added into the new cradle swarm.

The following is the framework of CPSO:

Algorithm 3 The CPSO Algorithm

```
1: Create two empty cluster lists $cur\_clst$ and $pre\_clst$ to
   store clusters of cradle swarm for current generation and
   previous generation.
2: Create one empty converge list $conver\_lst$ to record the
   best particles of converged sub-swarms
3: Set fitness evaluation counter $fes := 0$
4: Generate an initial cradle swarm
5: while $fes < Total\_Fes$ do
6:    Perform $cradle.global\_search(fes)$
7:    if $cradle.popsize > 1$ and $cradle.generations >$
      $num\_rgh$ then
8:       $pre\_clst := cur\_clst$
9:       $cur\_clst :=$ Rough_clustering(cradle)
10:   end if
11:   if same clusters exist in $cur\_clst$ and $pre\_clst$ then
12:      Produce temporary sub-swarms corresponding to
         those clusters and remove involved particles from
         the cradle swarm
13:      Evolve temporary sub-swarms using the global
         search method
14:   end if
15:   if the number of temporary sub-swarms $> 1$ then
16:      for Each temporary sub-swarm[$i$] do
17:         if sub-swarm[$i$].$generations > num\_ref$ then
18:            Perform Refining_clustering( sub-swarm[$i$])
19:            Remove temporary sub-swarm[$i$]
20:         end if
21:      end for
22:   end if
23:   for Each final sub-swarm $i$ do
24:      Perform sub-swarm[$i$].$local\_search(fes)$
25:   end for
26:   if there is overlapping among sub-swarms then
27:      Merge the overlapped sub-swarms
28:   end if
29:   Check convergence of sub-swarms, add the best parti-
      cles of converged swarms into $conver\_lst$ and remove
      them
30:   if an environment change is detected then
31:      Save the best particles of all sub-swarms into
         $conver\_lst$, and remove all sub-swarms
32:      Generate a new cradle swarm, then add the particles
         of $conver\_lst$ into the cradle swarm and empty
         $conver\_lst$
33:   end if
34: end while
```

IV. Experimental Study

A. Test problems

The performance of CPSO is tested on six problems generated by the benchmark GDBG proposed by Li et al.[13], [14]. There are seven change types of the system control parameters in the GDBG system. They are small step change, large step change, random change, chaotic change, recurrent change, recurrent change with noise and dimensional change. The framework of the seven change types are described as follows:

Framework of $DynamicChanges$:

switch(change type)
case small step:

$$\Delta\phi = \alpha \cdot \|\phi\| \cdot r \cdot \phi_{severity} \tag{6-1}$$

case large step:

$$\Delta\phi = \|\phi\| \cdot (\alpha \cdot sign(r) + (\alpha_{max} - \alpha) \cdot r) \cdot \phi_{severity} \tag{6-2}$$

case random:

$$\Delta\phi = N(0,1) \cdot \phi_{severity} \tag{6-3}$$

case chaotic:

$$\phi(t+1) = A \cdot (\phi(t) - \phi_{min}) \cdot (1 - (\phi(t) - \phi_{min})/\|\phi\|) \tag{6-4}$$

case recurrent:

$$\phi(t+1) = \phi_{min} + \|\phi\|(\sin(\frac{2\pi}{P}t + \varphi) + 1)/2 \tag{6-5}$$

case recurrent with noisy:

$$\phi(t+1) = \phi_{min} + \|\phi\|(\sin(\tfrac{2\pi}{P}t + \varphi) + 1)/2 + N(0,1) \cdot noisy_{severity} \tag{6-6}$$

case dimensional change:

$$D(t+1) = D(t) + sign \cdot \Delta D \tag{6-7}$$

where $\|\phi\|$ is the change range of ϕ, $\phi_{severity}$ is a constant number that indicates change severity of ϕ, ϕ_{min} is the minimum value of ϕ, $noisy_{severity} \in (0,1)$ is noisy severity in recurrent with noisy change. $\alpha \in (0,1)$ and $\alpha_{max} \in (0,1)$ are constant values, which are set to 0.04 and 0.1 in the GDBG system. A logistics function is used in the chaotic change type, where A is a positive constant between $(1.0, 4.0)$, if ϕ is a vector, the initial values of the items in ϕ should be different within $\|\phi\|$ in chaotic change. P is the period of recurrent change and recurrent change with noise, φ is the initial phase, r is a random number in $(-1,1)$, $sign(x)$ returns 1 when x is greater than 0, returns -1 when x is less than 0, otherwise, returns 0. $N(0,1)$ denotes a normally distributed one dimensional random number with mean zero and standard deviation one. ΔD is a predefined constant, which is defaulted to 1. If $D(t) = Max\_D, sign = -1$; if $D(t) = Min\_D, sign = 1$. Max_D and Min_D are the maximum and minimum number of dimensions.

The six test problems defined in [14] are: F_1 Rotation peak function, F_2 Composition of Sphere's function,F_3 Composition of Rastrigin's function, F_4 Composition of Griewank's function, F_5 Composition of Ackley's function and F_6 Hybrid Composition function.The parameters of the six problems are set as the same as in [14].

B. Performance Evaluation

There are total 49 test cases for the six test problems, for each test case, the average best, average mean, average worst values and STD are recorded, which are defined as in [14]:

$$Avg\_best = \sum_{i=1}^{runs} Min_{j=1}^{num\_change} E_{i,j}^{last}(t)/runs$$
$$Avg\_mean = \sum_{i=1}^{runs} \sum_{j=1}^{num\_change} E_{i,j}^{last}(t)/ (runs * num\_change)$$
$$Avg\_worst = \sum_{i=1}^{runs} Max_{j=1}^{num\_change} E_{i,j}^{last}(t)/runs$$
$$STD = \sqrt{\frac{\sum_{i=1}^{runs} \sum_{j=1}^{num\_change} (E_{i,j}^{last}(t) - Avg\_mean)^2}{runs * num\_change - 1}}$$

where $E^{last}(t) = |f(x_{best}(t)) - f(x^*(t))|$ and $x^*(t)$ is the global optimum at time t.

For the total 49 test cases, the overall performance of CPSO is calculated according to the method proposed in [14].

C. Parameters Setting

In the paper, the performance of CPSO is compared with the standard PSO (SPSO) and a simple genetic algorithm (SGA). Both SPSO and SGA use the restart with elitism scheme. That is, when an environment change is detected, the population is re-initialized and the best individual in the previous generation is replaced into the restarted population. For each test case, all the three algorithms were run 20 times.

In CPSO, the population size is set to $20 * n$, where n is the number of dimensions. For problem F_1, num_rgh and num_ref are set to 5 and 3 respectively. For F_2-F_6, they are set to 7 and 4. The acceleration constants η_1 and η_2 are both set to 1.7, the inertia weight ω linearly decreases from 0.6 in the first generation to 0.3 in the last generation for sub-swarm[i] as follows:

$$\omega = \omega_{max} - \frac{(\omega_{max} - \omega_{min}) sub\_swarm[i].iters}{sub\_swarm[i].iters + left\_iters} \tag{7}$$

where $left\_iters = (Total\_Fes - fes)/c\_particles$, $c\_particles$ is the current number of total particles in all sub-swarms and cradle swarm. $\omega_{max} = 0.6$, $\omega_{min} = 0.3$.

For SPSO, the acceleration constants η_1 and η_2 are both set to 1.49618 and the inertia weight $\omega = 0.729844$. The population size of SPSO and SGA are both 100. The crossover and mutation probabilities are set to 0.8 and 0.02 respectively in SGA.

D. Results and Discussions

Table I-Table VII present the results of the average best, average mean, average worst values and the standard variance (STD) of each test case of CPSO, SGA, and SPSO. Table IV-D shows the performance of CPSO, SGA, and SPSO on each test case.

TABLE I
ERROR VALUES ACHIEVED FOR PROBLEMS F_1 ON 10 PEAKS

	Errors	T_1	T_2	T_3	T_4	T_5	T_6	T_7
CPSO	Avg best	1.054e-7	5.214e-8	4.306e-8	9.721e-7	2.561e-7	4.325e-6	5.036e-9
	Avg worst	1.244	27.12	28.15	3.239	21.72	26.55	35.52
	Avg mean	0.03514	2.718	4.131	0.09444	1.869	1.056	4.54
	STD	0.4262	6.523	8.994	0.7855	4.491	4.805	9.119
SGA	Avg best	4.01e-5	4.295e-5	5.543e-5	0.0001799	0.0001004	0.0002894	6.234e-6
	Avg worst	43.2	52.08	45.47	75.39	40.23	80.31	42.76
	Avg mean	5.609	10.08	13.13	21.22	7.899	29.25	12.45
	STD	9.349	13.22	13.87	21.88	9.406	25.68	14.6
SPSO	Avg best	0	0	0	0	0	0	0
	Avg worst	31	48.23	43.28	72.77	35.77	78.92	46.77
	Avg mean	5.669	10.24	11.73	21.89	6.731	32.01	12.8
	STD	7.729	12.62	13.59	20.15	8.75	25.63	14.04

TABLE II
ERROR VALUES ACHIEVED FOR PROBLEMS F_1 ON 50 PEAKS

	Errors	T_1	T_2	T_3	T_4	T_5	T_6	T_7
CPSO	Avg best	2.447e-6	2.061e-7	9.888e-7	4.353e-6	2.121e-6	9.033e-5	4.169e-6
	Avg worst	4.922	22.08	25.65	1.974	9.606	22.08	27.9
	Avg mean	0.2624	3.279	6.319	0.125	0.8481	1.482	6.646
	STD	0.9362	5.303	7.442	0.3859	1.779	4.393	7.94
SGA	Avg best	0.000143	0.0001435	0.0002528	0.0004217	0.0005458	0.001029	0.0001081
	Avg worst	40.16	44.75	47.84	70.65	28.03	78.24	44.34
	Avg mean	7.614	11.3	15.24	17.93	5.293	34.93	15.68
	STD	9.754	11.26	13.04	19.04	6.186	26.54	12.42
SPSO	Avg best	0	0	0.0001245	0	0.000989	7.791e-013	0
	Avg worst	33.32	46.08	45.33	69.84	28.23	78.32	44.03
	Avg mean	7.95	12.29	14.89	20.96	5.426	36.27	15.94
	STD	8.162	11.55	12.5	19.02	6.348	26.24	12.02

TABLE III
ERROR VALUES ACHIEVED FOR PROBLEMS F_2

	Errors	T_1	T_2	T_3	T_4	T_5	T_6	T_7
CPSO	Avg best	9.377e-05	7.423e-05	4.651e-05	1.121e-05	7.792e-05	0.0001087	2.978e-07
	Avg worst	19.26	144.1	158.3	10.18	320.7	26.08	30.44
	Avg mean	1.247	10.1	10.27	0.5664	25.14	1.987	3.651
	STD	4.178	35.06	33.45	2.137	64.25	5.217	6.927
SGA	Avg best	0.001909	0.003022	0.005739	0.002071	0.009138	0.003432	0.002784
	Avg worst	150.5	565.5	543.6	124.8	511	289.4	460.5
	Avg mean	33.05	182.9	128.5	32.85	191.7	43.25	40.59
	STD	53.75	218.9	188.7	35.12	200.6	69.84	86.74
SPSO	Avg best	1.016e-013	0	4.334e-014	7.523e-014	0	0	0
	Avg worst	272.3	561	539.4	279.3	515.6	541.6	469.9
	Avg mean	45.79	186.9	135.8	53.57	186.5	73.34	61.13
	STD	59.34	212.7	185.4	60.58	198.1	99.96	99.49

From the results, it can be seen that CPSO performs much better than the other two algorithms on most test cases over all change types except on F_3. For F_3, the results of CPSO are worse than that of SGA except with the small change type. SGA may benifit from its better diversity than CPSO. The interesting thing is that the average best result of SGA on F_3 with recurrent change is better than CPSO, but the SGA's score obtained in this case is much less than CPSO. This shows that CPSO converges faster than SGA due to its local search method.

By observing the results, it can be seen that the challenge of different change types is quite different. The small step change is the easiest for algorithms on most test cases. The large step and chaotic change bring in the biggest challenge on most test cases. The dimensional change is also difficult to optimize for algorithms.

The results show that different problems have a different difficulty for algorithms. Because of F_1's smooth fitness landscape, F_1 is the simplest one to optimize. The composition problems are difficult for algorithms to get the global optima. F_3 is the most difficult one among all the test problems.

V. CONCLUSIONS

The paper proposes a clustering particle swarm optimization (CPSO) algorithm for dynamic optimization problems. CPSO employs a hierarchical clustering method to track multiple peaks based on a nearest neighbor global search strategy. A new learning mechanism of the global best particle is introduced to find the near optimal solutions in

TABLE IV
ERROR VALUES ACHIEVED FOR PROBLEMS F_3

	Errors	T_1	T_2	T_3	T_4	T_5	T_6	T_7
CPSO	Avg best	0.003947	126.2	42.89	7.909e-005	228.5	4.356	0.9334
	Avg worst	711.2	1008	966.1	1204	974.2	1424	1011
	Avg mean	137.5	855.1	765.9	430.6	859.7	753	653.7
	STD	221.6	161	235.8	432.2	121.5	361.7	334
SGA	Avg best	0.009432	0.3146	2.045	0.5873	36.15	0.075	0.01136
	Avg worst	786.1	1036	991.7	1286	970.5	1380	1006
	Avg mean	158.1	638.7	573.9	419.5	741.9	491.7	499.5
	STD	264.5	399.6	399.8	444.2	278.8	464.3	399.8
SPSO	Avg best	1.427	211.4	20.9	3.82	13.59	3.782	2.481
	Avg worst	864.1	1068	1024	1396	990.2	1509	1071
	Avg mean	553.6	900.8	827.1	709	829.1	803.5	715.4
	STD	298.1	148.8	212.6	385.8	186.7	375	334.9

TABLE V
ERROR VALUES ACHIEVED FOR PROBLEMS F_4

	Errors	T_1	T_2	T_3	T_4	T_5	T_6	T_7
CPSO	Avg best	6.36e-005	0.0001868	0.000103	9.346e-006	0.000407	8.616e-005	3.31e-006
	Avg worst	29.38	459.8	389.4	14.62	481	63.06	93.32
	Avg mean	2.677	37.15	36.67	0.7926	67.17	4.881	7.792
	STD	7.055	99.43	97.18	2.775	130.3	15.39	19.21
SGA	Avg best	0.002697	0.003439	0.007537	0.001855	0.04842	0.003322	0.007408
	Avg worst	296.5	643.3	624.3	376.2	590.9	595.3	616.9
	Avg mean	45.92	272.9	230.1	52.76	335.5	57.38	93.45
	STD	80.15	270.7	251.2	96.98	223.7	116.6	173.7
SPSO	Avg best	0	0	0	0.3056	0	0	0
	Avg worst	376.3	656.1	612.9	460.3	576.1	684.4	601.1
	Avg mean	55.05	289.7	223.6	73.85	285	98.15	92.21
	STD	92.64	263	245.1	104.8	228.1	148.4	150.4

TABLE VI
ERROR VALUES ACHIEVED FOR PROBLEMS F_5

	Errors	T_1	T_2	T_3	T_4	T_5	T_6	T_7
CPSO	Avg best	0.0001584	0.0003224	0.0003337	4.85e-006	0.0001377	0.0002077	2.052e-006
	Avg worst	25.41	31.76	27.77	26.66	63.2	42.54	103.2
	Avg mean	1.855	2.879	3.403	1.095	7.986	4.053	6.527
	STD	5.181	6.787	6.448	4.865	13.81	8.371	22.8
SGA	Avg best	0.006832	0.007609	0.00571	0.003871	0.008463	0.005129	0.005733
	Avg worst	80.54	82.92	75.17	89.64	64.14	89.61	78
	Avg mean	27.99	29.57	25.4	33.96	24.42	31.77	23.19
	STD	24.23	25.31	21.92	30.98	19.39	30.97	20.76
SGA	Avg best	5.857e-007	0	0	0	0	0	0
	Avg worst	554.7	500.4	360.5	740	94.04	945	869.7
	Avg mean	62.22	58.85	44.51	91.95	29.03	116.9	73.06
	STD	104	99.23	67.86	149.7	22.24	193.1	152.7

a local promising region. Six test problems are used to test the performance of the proposed algorithm. The numerical experimental results show CPSO's good performance on these test problems.

Though the clustering method is effective to generate sub-swarms, it is still difficult to get the accurate sub-swarms, particularly for the situation that only one particle covers a single peak. More work should be done to solve this problem.

REFERENCES

[1] T. M. Blackwel. Particle swarms and population diversity II: Experiments. *Proc. of the 2003 Genetic and Evol. Comput. Workshops*, pp. 108-112, 2003.

[2] T. M. Blackwell and J. Branke. Multi-swarm optimization in dynamic environments. *Applications of Evolutionary Computing*, LNCS 3005, pp. 489-500, 2004.

[3] T. M. Blackwell and J. Branke. Multiswarms, exclusion, and anti-convergence in dynamic environments. *IEEE Transactions on Evolutionary Computation*, **10**(4): 459-472, 2006.

[4] J. Branke. Memory enhanced evolutionary algorithms for changing optimization problems. *Proc. of the 1999 Congr. on Evol. Comput.*, vol. 3, pp. 1875-1882, 1999.

[5] J. Branke, T. Kaußler, C. Schmidth, and H. Schmeck. A multi-population approach to dynamic optimization problems. *Proc. 4th Int. Conf. on Adaptive Computing in Design and Manufacturing*, pp. 299-308, 2000.

[6] R. Brits, A. P. Engelbrecht and F. van den Bergh. Solving systems of unconstrained equations using particle swarm optimization. *Proc. IEEE Conf. Syst,. Man, Cyber.*, pp. 102-107, 2002.

[7] R. Brits, A. Engelbrecht, and F. van den Bergh. A Niching Particle Swarm Optimizer. *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL'02)*, vol. 2, pp. 692–696, 2002.

[8] M. Clerc and J. Kennedy. The particle swarm: Explosion, stability and convergence in a multi-dimensional complex space. *IEEE Trans. on Evolutionary Computation*, **6**: 58-73, 2002.

TABLE VII

Error Values Achieved for Problems F_6

	Errors	T_1	T_2	T_3	T_4	T_5	T_6	T_7
CPSO	Avg best	0.0001693	0.000126	0.0006566	1.28e-005	0.001835	0.0002852	0.0002053
	Avg worst	37.79	258.5	504.8	131.8	628.8	265.7	424.5
	Avg mean	6.725	21.57	27.13	9.27	71.57	23.67	32.58
	STD	9.974	63.51	83.98	24.23	160.3	51.55	76.9
SGA	Avg best	0.01314	0.004445	0.009612	0.1484	0.009385	0.005041	0.007232
	Avg worst	247.2	824.8	783.6	309.6	785	530.1	697.2
	Avg mean	39.41	138.6	98.51	53.53	170.1	52.1	51.34
	STD	65.84	254.3	208.8	100.2	274.6	87.99	135.1
SPSO	Avg best	1.457	9.832e-014	9.699e-014	4.941e-012	0	0	1.169e-006
	Avg worst	546.3	842.4	806.1	682.1	817.1	748.6	710
	Avg mean	71.15	158.7	140.3	120.7	162.8	113.8	101.7
	STD	118.1	260.8	240.7	173.3	275	164.6	173.7

TABLE VIII

Algorithm Overall Performance

		$F1(10)$	$F_1(50)$	F_2	F_3	F_4	F_5	F_6
CPSO	T_1	0.942163	0.940825	0.727937	0.263052	0.687955	0.664818	0.556384
	T_2	0.892462	0.887899	0.574953	0.0183243	0.470139	0.611798	0.439927
	T_3	0.868731	0.837547	0.580325	0.0375368	0.489701	0.602617	0.431848
	T_4	0.976683	0.975418	0.899955	0.276901	0.883281	0.874479	0.630821
	T_5	0.889075	0.917639	0.568815	0.0271835	0.462767	0.608712	0.414599
	T_6	0.881828	0.873017	0.643585	0.0345534	0.568689	0.538666	0.358982
	T_7	0.85684	0.829573	0.65473	0.07386	0.572144	0.588575	0.41351
	Mark	0.0903325	0.0897809	0.106369	0.016963	0.0946551	0.103043	0.0745976
GA	T_1	0.871751	0.843782	0.371132	0.262886	0.336349	0.383906	0.333081
	T_2	0.807799	0.794346	0.253051	0.111445	0.21354	0.36591	0.29161
	T_3	0.746394	0.729673	0.317967	0.161118	0.257419	0.405055	0.336495
	T_4	0.672787	0.708963	0.326789	0.15134	0.292427	0.308862	0.260051
	T_5	0.831102	0.874166	0.298643	0.0723514	0.187631	0.508448	0.38096
	T_6	0.585837	0.524853	0.294985	0.139062	0.297623	0.332803	0.247792
	T_7	0.76308	0.717596	0.395579	0.1558	0.336565	0.409334	0.396201
	Mark	0.0753658	0.0743127	0.0510309	0.0240497	0.0434248	0.061869	0.050739
SPSO	T_1	0.851997	0.826273	0.31644	0.0989168	0.325098	0.299304	0.282714
	T_2	0.798678	0.777322	0.228137	0.0178981	0.185257	0.303377	0.265428
	T_3	0.767943	0.729666	0.290895	0.0334057	0.2586	0.363386	0.289041
	T_4	0.647215	0.659869	0.242093	0.049537	0.24865	0.231125	0.182946
	T_5	0.850754	0.874163	0.314564	0.040489	0.239287	0.474711	0.391785
	T_6	0.543241	0.504159	0.220579	0.0432205	0.213175	0.220548	0.180305
	T_7	0.737354	0.709785	0.334463	0.0744435	0.287422	0.333918	0.294171
	Mark	0.074271	0.0726696	0.0440564	0.00799431	0.0398804	0.0507615	0.04292
Performance(sum all mark and multiply by 100) CPSO: 57.5742 GA: 38.0792 SPSO: 33.2553								

[9] Eberhart, R. C. and J. Kennedy. A new optimizer using particle swarm theory. *Proc. of the 6th Int. Symp. on Micro Machine and Human Science*, pp. 39-43, 1995.

[10] Kennedy, J. and R. C. Eberhart. Particle Swarm Optimization. *Proc. of the 1995 IEEE Int. Conf. on Neural Networks*, pp. 1942-1948, 1995.

[11] J. Kennedy, Stereotyping: Improving particle swarm performance with cluster analysis, in *Proc. Congr. Evol. Comput.*, 2000, pp. 1507-1512.

[12] Kennedy, J. and R. C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann Publishers.(2001).

[13] C. Li and S. Yang. A generalized approach to construct benchmark problems for dynamic optimization.*Proceedings of the 7th Int. Conf. on Simulated Evolution and Learning*, 2008. Springer.

[14] C. Li, S. Yang, T. T. Nguyen, E. L. Yu, X. Yao, Y. Jin, H.-G. Beyer, and P. N. Suganthan, Benchmark Generator for CEC'2009 Competition on Dynamic Optimization. *Technical Report 2008*, Department of Computer Science, University of Leicester, U.K., 2008.

[15] S. Janson and M. Middendorf. A hierachical particle swarm optimizer for dynamic optimization problems. *Applications of Evolutionary Computing*, LNCS 3005, pp. 513-524, 2004.

[16] D. Parrott and X. Li. A particle swarm model for tracking multiple peaks in a dynamic environment using speciation. *Proc. of the 2004 IEEE Congress on Evolutionary Computation*, pp. 98-103, 2004.

[17] K. E. Parsopoulos and M. N. Vrahatis. Recent approaches to global optimization problems through particle swarm optimization. *Natural Computing*, **1**(2-3): 235-306, 2002.

[18] R. Poli, J. Kennedy, and T. Blackwell. Particle swarm optimization: An overview. *Swarm Intelligence* , **1**(1): 33-58, 2007.

[19] F. van den Bergh. An analysis of particle swarm optimizers, *Ph.D dissertation*, Dept.Comput. Sci., Univ.Pretoria, Pretoria, Gauteng, South Africa, 2002.

[20] F. van den Bergh, A. P. Engelbrecht. A Cooperative approach to particle swarm optimization. *IEEE Trans. on Evol. Comput.*, **8**(2): 225-239, 2004.

[21] H. Wang, D. Wang, and S. Yang. Triggered memory-based swarm optimization in dynamic environments. *Applications of Evolutionary Computing*, LNCS 4448, pp. 637-646, 2007.