A New Differential Evolution Algorithm with Dynamic Population Partition and Local Restart

Yuan-long Li and Jun Zhang (Corresponding Author)
Department of Computer Science, Sun Yat-sen University
Key Laboratory of Digital Life, Ministry of Education
Key Laboratory of Software Technology, Education Dept. of Guangdong Province, P.R. China
junzhang@ieee.org

ABSTRACT

This paper will introduce a new differential evolution (DE) algorithm called DE/cluster. DE/cluster applies a simple hierarchical clustering model to mine the distribution information of the DE population every K generations to make a dynamic partition of the population. One special cluster formed by the single-individual clusters will use a slower convergence mutation strategy to do the global search. The other clusters will use more greedy searching strategy to do the local search. As long as the subpopulations may be trapped by local minima, the "dead" state is defined for a cluster and clusters will be checked in every generation and the "dead" clusters will be restarted in the current searching range. This local restart strategy can make the performance of DE/cluster even be better than DE/rand on some multimodal test functions that are not linearly separable. The DE/cluster algorithm is tested on a test suite with 24 functions and it shows promising performance compared with the current best DE variants.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]:Problem Solving, Control Methods and search-Heuristic methods; G.1.6 [Numerical Analysis]: Optimization-Global optimization

General Terms: Algorithms.

Keywords

Differential Evolution; Cluster Analysis; Restart Strategy; Multiple Populations; Evolution Computing.

1. Introduction

Differential Evolution algorithm [1]-[2] is one of the most promising continuous optimization evolutionary algorithms. It has been studied for the past decades since its first publish in 1995. In general DE algorithm, for every generation, a population is maintained and updated by the mutation operator, the crossover operator and the selection operator in the mentioned order. In each generation, every individual of the population will be taken as a target vector once. For a target vector *i*, in the simplest cases, three distinct individuals r_1, r_2 and r_3 from the population are selected randomly, and then a mutant vector is generated in the following way:

$$v_i = x_{r_1} + F \cdot (x_{r_2} - x_{r_3}), \tag{1}$$

where *F* is a scale factor.

After the mutant vector v_i is created, it will be combined with the target vector x_i to generate a trial vector u_i by the crossover operator. The most common crossover operator used in DE is the binary crossover which works as (2):

$$u_i(j) = \begin{cases} v_i(j) & if\ j == jrand \,\|\, rand < CR \\ x_i(j) & otherwise \end{cases}. \tag{2}$$

Where *CR* is the crossover rate and *jrand* is an index randomly chosen from the *D* dimensions to ensure that at least one component of u_i is come from v_i (which makes u_i different with x_i whatever the value of *CR* is).

The trial vector u_i will be evaluated and compared with the target individual x_i and the better one will enter the next generation.

The mutation strategy of DE algorithm is very important for its performance. In the mutation operation as shown in (1), vector x_{r_1} is called the base vector, while $x_{r_2} - x_{r_3}$ is called the differential vector. Thus a mutant vector of DE can be defined as the linear combination of the base vector and the differential vector. There are many different ways to choose the base vector and the number of the differential vectors. DE/current-to-best algorithm applies a base vector generated by the linear combination of the best vector and the target vector.

Currently the study of DE has mainly focused on the following aspects. The first is the parameter adaptation [5] for that it is critical for DE algorithm to perform well on different kinds of test functions. But something new has been proposed to further enhance the performance of DE in the past year (2010). Among these new ideas, DE with a mutating strategy pool (SaDE[6]) tries to use different mutation strategies in the evolution procedure. In SaDE, each individual will choose a mutation strategy according to a probability which is adapted during the evolution procedure. The other new idea of DE recently is trying to incorporate the local guidance information

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
GECCO'11, July 12–16, 2011, Dublin, Ireland.
Copyright 2011 ACM 978-1-4503-0557-0/11/07...$10.00.

in DE. In JADE[8], the author modified the guiding individual "best" in the "DE/current-to-best" strategy from "the global best of the population" to "some individuals which are at the top *p*% of the current population". This can make more individuals the guiding individuals with a larger *p* setting. Another DE variant proposed recently with similar idea is DEGL[7]. DEGL applies a "local best" individuals concept and uses these individuals to guide the mutation. A local best individual is indentified in a storage neighborhood structure (for individual *i*, its guiding vector is the best of individuals indexed within the range [*i-t*, *i+t*]). DEGL actually do an implicit partition of the current population and using the best individuals in each subset to guide the evolution.

With the introduction on the most recent studies on DE, we think that there are still some aspects which may be further improved:

a) In SaDE, the mutation strategy pool does not contain the fast converging mutation strategy "DE/best", which may greatly accelerate the searching speed of the algorithm if it is well restricted. Another problem is that different individuals may need different kinds of mutation strategy while in SaDE the probability to choose a mutation strategy is the same for every individual.

b) In the idea of DEGL or JADE, the size of the subset to be chosen for the selection of a "local best" or the top proportion to be chosen as the guiding vector is locked and static during evolution. Proper setting of the size for different kinds of problems may be very different.

c) The topology of the subset partition of DEGL has been chosen in a way which actually considers no information of the distribution of the current population in the searching space.

With the above analysis, in this paper we will introduce a new DE algorithm called DE/cluster which tries to use the cluster information of the current population to dynamically partition the population and choose proper mutation strategies for different individuals.

The simple hierarchical clustering algorithm is applied on the population every *K* generations in DE/cluster for which the max size of a cluster is limited. (Clustering in a bottom-up way, small clusters will be combined together until the size of the maximum cluster will exceed a predefined maximum size). The clustering results will be further modified by combining all the single individual clusters and make this cluster a special cluster. This cluster will apply the slower convergence searching strategy DE/current-to-best while the other clusters will apply the more greedy searching strategy DE/best. The best individual in each cluster will lead the search of its cluster. Clusters with some small number of individuals will get enough individuals to generate differential vectors from a "pool" which stores some additional individuals from the parent population. In this way, the clustering partition results are used in DE/cluster.

In DE/cluster a local restart strategy is also proposed. The local restart strategy will first check all the subpopulations with proper size and if a subpopulation is not the best but highly converged, it will be considered as "dead" and will reborn in the current searching range. With the local restart strategy, DE/cluster is even able to perform better than DE/rand on some rotated multimodal functions that are not linearly separable (for which the algorithm are greatly relying on the mutation strategy but not the *CR* adaptation).

Based on the above idea, the DE/cluster algorithm is proposed in this paper. The DE/cluster algorithm achieves the balance of the global search and local search by imposing different searching strategies to different subpopulations. Cluster analysis provides the information to partition the population dynamically and different subpopulation can evolve independently with proper mutation strategies. The local restart strategy can make the algorithm more robust.

The following part of this paper will be organized as following. In section II, the DE/cluster algorithm will be introduced. In section III, some experiments of DE/cluster will be carried out to prove its performance. Section IV concludes the whole paper.

2. DE/cluster Algorithm

To achieve the balance of the searching speed and the diversity of the population, it is important to make the base vectors contain both the position information of the best individual and the whole distribution of the population in the searching space. This is why [7] applies an implicit partition of the population and make them evolve with the base vector set as the best individuals in every subpopulation. But how to partition the population is an important problem.

Here we are going to using the cluster information to solve this problem. Cluster information of the current population could be useful for detecting the type of the function in the current searching range. The DE population can adapt itself to the shape of the function, so it is possible to use the cluster analysis to mining this information and use it to get a dynamic partition of the current population. Through some simple cluster analysis tools, the population can be divided into different clusters, and every individual in the population will select the best individual of its cluster as the guiding vector (shows in Fig. 1).

Here the subsets will evolve for more than one generation until the next time of clustering to reduce the cost of the cluster analysis, while the re-clustering for every *K* generations makes it possible to find new clusters. Still there are some other detailed problems in the current showed framework and the following section will introduce the detailed operations and some strategies to solve the problems encountered.

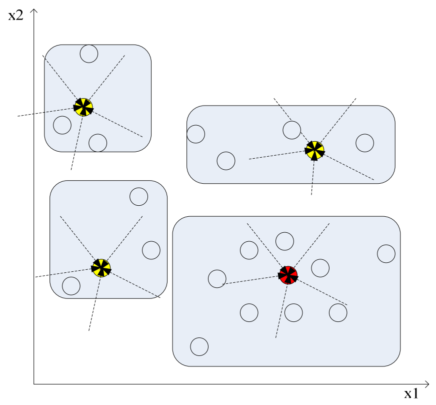

Figure 1. Clusters of the population in DE/cluster. Every cluster is guided by its best individual. Clusters may of different size.

2.1 Basic Cluster Analysis in DE/cluster

This part will introduce the basic cluster analysis applied to the current population. As has been mentioned, this analysis will be executed in every *K* generations, where *K* is a parameter which controls the frequency of the cluster analysis. The data to be clustered are the *NP* individuals in the current population. The hierarchical clustering method is applied here to do the analysis. Steps to do the analysis are as follows:

a) Calculate the Euclidean distances between every two individuals.

b) With the distance matrix and the function value of every individual, the linkage tree for clustering is created in a bottom-up way. At the beginning, every individual is set as a cluster and set itself as its representative individual. Then choose the two clusters with the smallest in-between distance and combine them as one cluster, and set the representative individual as the better one of the representative individuals of the original two clusters. The distance between two clusters is calculated as the distance of their representative individuals as following:

$$d(G_i, G_j) = d(I(G_i), I(G_j)) \cdot \quad (3)$$

Where G_i stands for the *i*-th cluster, $I(G_i)$ stands for the representative individual of the *i*-th cluster.

c) The bottom-up building iteration procedure will be stopped if the current combination will generate a cluster that is large than a predefined largest size *S*. This setting makes the population never be clustered as a single cluster.

By now, the basic cluster analysis has been done and the clusters will be further processed to be used to evolve independently.

2.2 Further Processing for Independent Evolution

The results provided by the basic cluster analysis are far from direct independent evolution. Some of the clusters may be too small to evolve independently, because the evolution of DE population needs enough individuals to generate differential vectors (at least 3, but more vectors should be used to provide some different differential vectors). So it is urgent to solve this problem. This problem could be taken as the defect of the dynamical partition of the population, compared to the static partition ways.

To solve this problem, we need to analyze the reason why some clusters are too small. It could be explained as that some of the individuals are more centralized than the others, which is possible because the DE evolution procedure will make the population converge to an extreme point. From this point of view, here the small clusters are handled in the ways as following.

a) The clusters each with only one individual. For these clusters, it could be distributed far away from the other individuals. In this case the single individual clusters are actually maintaining a lot of diversity information of the current population. To proper handle this case, we will combine all these single-individual-clusters as ONE large cluster and if the current best individual is not belonging to this cluster, this cluster will be specially marked as the *SPEX* cluster, because of its key roles in maintaining the diversity of the current population. This *SPEX* cluster will apply more global searching strategy than the other clusters.

b) The clusters with more than one individual but still too small to generate enough different differential vectors. Set the minimum number of individuals to generate enough differential vectors as *M*1. For the clusters with not enough individuals, we need to find some other individuals which can help generate proper differential vectors. DE/cluster maintains a pool for every cluster to generate differential vectors. The pool is updated in every generation before the evolution beginning. Firstly the number of the individuals of each cluster is checked, and if the number is no smaller than *M*1, the pool for this cluster will be all the individuals in this cluster. If it is smaller than *M*1, the individuals in the cluster will be put into the new pool first and the nearest cluster pool of it in the last generation is found and the individuals in that pool will be added into the new cluster until its number of individuals is larger than *M*2. The best individual in the pool will be taken as the representative individual and the distance between clusters is calculated as the distance of their representative individuals.

Then the cluster analysis module provides the following information:

a) For every individual *i* in the current population, the cluster it belonging to *T*(*i*).

b) For every cluster *k*, the best individual *best*(*k*).

c) For every cluster *k*, its differential vector pool *Pool*(*k*).

d) For the special marked cluster, it is marked by a special index *SPEX*.

2.3 Evolution Rules for Every Cluster in DE/cluster

In the last part the cluster analysis has provided some useful information. This section will show how the information is used, mainly about the different iteration rules for different kinds of clusters and the interaction among clusters.

2.3.1 Independent Evolution of Every Cluster

The population iteration is still working like original DE iterations, but with different iteration strategies for different kinds of clusters. There are two kinds of clusters generated by the cluster analysis module. Overall the population iteration procedure is as follows:

For every individual *i* in the current population, according to the cluster *T*(*i*) it belonging to,

a) If $T(i) \neq SPEX$

Then the individual will use the DE/best/2 as the iteration strategy, i.e.:

$$v_i = x_{best(T(i))} + F*(x_{r2} - x_{r3}) + F*(x_{r4} - x_{r5}) \,. \quad (4)$$

Where x_{r2}, x_{r3}, x_{r4} and x_{r5} are individuals randomly selected from the differential pool of cluster *T*(*i*), $x_{best(T(i))}$ stands for the best individual of cluster *T*(*i*).

b) If $T(i) == SPEX$

When the pool for *SPEX* cluster has enough individuals:

$$v_i = x_i + F*0.6*(x_{best(SPEX)} - x_i) + F*(x_{r2} - x_{r3}) \,. \quad (5)$$

Otherwise the iteration strategy is like this:

$$v_i = x_i + F*(-x_{best(SPEX)} + x_i) + F*(x_{r2} - x_{r3}) \,. \quad (6)$$

Noticed that when there not enough individuals in the pool for some cluster (less than *M*1, at the beginning stage of the evolution); the differential vectors are generated from the whole population.

Overall we can see that different DE strategies have been applied in DE/cluster to gain advantage of their different convergence characteristics. The searching strategy for the *SPEX* cluster will be used to do the exploring job for the current searching range, while the other clusters apply fast convergence DE/best strategy to exploit some local minima. Here we use different DE strategies in a way which is different from [6].

2.3.2 "Dead" Clusters and Local Restart

The independent evolution of every cluster may cause a serious problem that some clusters are trapped in local minima. Here we will going to define the "dead" state for a cluster which shows barely promising to find better solution than the current global best individual. As no information of the best function value is known, the quality of a cluster can only be defined by its comparison results with the other clusters. So here we are going to define the "dead" state for a cluster by the interaction among the clusters of current population. If a cluster is converging, the difference among the fitness of the individuals in the cluster will be smaller and smaller. If proper conditions can be defined to find out such kind of local trapped cluster, then these clusters should be reborn to release the sources for other search space.

The "dead" state will be used to define such kind of state of a cluster.

Definition: Cluster *k* is defined as "dead" if

$$\begin{aligned}
&k \neq T(best)\ \&\ \& \\
&k \neq SPEX\ \&\ \& \qquad\qquad (7)\\
&pcount(k) >= (NP - pcount(T(best)))/(cc-1)\ \&\ \& \\
&Fstd(k) <= 0.1 * |Fbest - Fst(k)|
\end{aligned}$$

Where *k* is used to indicate the cluster to be checked; the best cluster is indicated by *T*(*best*); the number of clusters is indicated by *cc*; the standard deviation of the fitness value of cluster *k* is indicated by *Fstd*(*k*); the fitness value of the best individual of cluster *k* is indicated by *Fst* (*k*); the best fitness value of the current population is indicated by *Fbest*; The number of the individuals in cluster *k* is indicated by *pcount*(*k*); the total number of the individuals in the population is indicated as *NP*.

In (7), line 1 and line 2 are used to exclude the best cluster and the *SPEX* cluster; Line 3 is used to make sure that the size of the cluster is above the average size of all the clusters except the best cluster; Line 4 is used to exclude cluster which has gained little progress in the last generation, with the threshold value set as 0.1. The standard deviation of fitness value in a cluster can show its progress in the last generation and if the progress is too small compared to the gap between the best fitness value in this cluster and the global best, it is reasonable to judge it as "dead".

The dead state defined above is just an empirical conclusion, so the "dead" cluster should be restarted in a soft way which will still be able to search the space where it is concerned dead. The local restart rules are as follows:

a) Find the current lower bound *CLB* and upper bound *CUB* in every dimension of the whole population.
b) For every individual in the dead cluster, with a probability of 0.9 the new individual will be generated by the DE/rand/2 strategy with the base vector randomly selected not belonging to the best cluster. Otherwise for its every dimension, 50% probability its new value will be uniformly randomly generated between [*CLB*, *CUB*],and another 50% it will be generated out of the current searching range by a Gaussian distribution with a standard deviation of *CUB-CLB*.

Here the restarting are only near the current searching space to make sure that the population will converge eventually. Immediate cluster analysis will be carried out when dead clusters have been restarted. Then new clusters may be found, which means the population will try to exploit new minima. Total iteration procedure of the population iteration with cluster analysis is shown in Fig. 2.

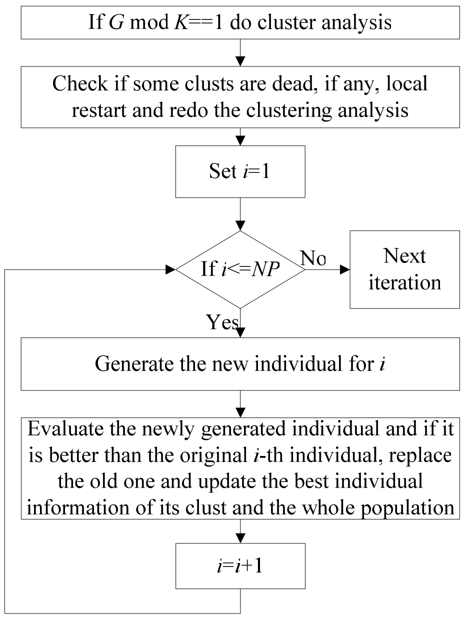
If G mod K==1 do cluster analysis
Check if some clusts are dead, if any, local restart and redo the clustering analysis
Set i=1
If i<=NP
No
Next iteration
Yes
Generate the new individual for i
Evaluate the newly generated individual and if it is better than the original i-th individual, replace the old one and update the best individual information of its clust and the whole population
i=i+1

Figure 2. Procedure of the DE/cluster main loop

2.4 Adaptation of F and CR

In DE algorithm, the setting of parameters *F* and *CR* is important to achieve good performance. Many parameters adaptation methods have been proposed recently [5][6][8]. In DE/cluster, we adopted the *F* and *CR* adaptation strategy of [6]. The adaptation of *F* is just use a random number generated by a preset Gaussian distribution *N*(*Fmean*, *Fstd*), while the *CR* values are generated by a dynamically adapted Gaussian distribution *N*(*CRmean*, *CRstd*). The value of *CRmean* is updated by the mean value of the *CR*s performed well in the last *LP* generations while *CRstd* is set as a static value. For detailed adaptation procedure of *F* and *CR*, please look for [6].

```
For each dimension j of new individual i:
  If rand<CR | j==jrand
    x[i][j] =new value generated by the base vector and differential vector;
  Else if rand<Alpha
    If rand<0.9
      x[i][j]=Pool(k,t,j);// K~=T(best) is a clust, t is a vector in pool k, both are
                                    randomly chosen.//GA crossover
    Else
      x[i][j]=uniformly random number In [LB(j),UB(j)] //GA mutation
    End
  Else
    x[i][j]=original value of j-th dimension
  End
End
```

Figure 3. Crossover strategy in DE/cluster

The *CR* adaptation strategy applied in DE/cluster has been slightly modified. The crossover in DE is actually a way to improve the quality of the solution by combining different values from different individuals when the variables in different dimensions are independent. In the standard DE algorithm crossover is only done between the farther individual and its child. In DE/cluster this is further extended by a GA [3] crossover operator, in which the individual not only crossover with the newly generated individual, but also with the individuals in the current population. This is based on the idea that the other individuals in the current population may have some "good values" which this individual does not. From this motivation the GA mutation strategy is also introduced to DE/cluster. Overall the crossover strategy of DE/cluster is shown in Fig. 3.

Table 1. Test Functions

F1: Sphere : shifted	$f_1(x)=\sum_{i=1}^{D} x_i^2$
F2: Schwefel 1.2 shifted	$f_2(x)=\sum_{i=1}^{D}(\sum_{j=1}^{i} x_j)^2$
F3: Rosenbrock	$f_3(x)=\sum_{i=1}^{D-1}(100(x_i^2-x_{i+1})^2+(x_i-1)^2)$
F4: Schwefel 1.2 shifted with noise:	$f_4(x)=(\sum_{i=1}^{D}(\sum_{j=1}^{i} x_j)^2)*(1+0.4\|N(0,1)\|)$
F5: Ackley shifted: $x=x-o$	$f_5(x)=-20*\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2})-\exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i))+20+e$
F6: Ackley shifted rotated: $x=M(x-o)$	$f_6(x)=-20*\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2})-\exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i))+20+e$
F7: Griewank shifted:	$f_7(x)=\sum_{i=1}^{D}\frac{x_i^2}{4000}-\prod_{i=1}^{D}\cos(x_i/\sqrt{i})+1$
F8: Griewank shifted rotated: $x=M(x-o)$	$f_8(x)=\sum_{i=1}^{D}\frac{x_i^2}{4000}-\prod_{i=1}^{D}\cos(x_i/\sqrt{i})+1$
F9: Rastrigin shifted:	$f_9(x)=\sum_{i=1}^{D}(x_i^2-10\cos(2\pi x_i)+10)$
F10: Rastrigin shifted rotated: $x=M(x-o)$	$f_{10}(x)=\sum_{i=1}^{D}(x_i^2-10\cos(2\pi x_i)+10)$
F11: Non-continuous Rastrigin shifted:	$f_{11}(x)=\sum_{i=1}^{D}(y_i^2-10\cos(2\pi y_i)+10)$, $y_i=\begin{cases}x_i, & \|x_i\|<1/2\\ round(2x_i)/2, & \|x_i\|\ge 1/2\end{cases}$
F12: Schwefel	$f_{12}(x)=418.9829*D-\sum_{i=1}^{D}x_i\sin(\|x_i\|^{1/2})$
F13: Schwefel's problem 2.22	$f_{13}(x)=\sum_{i=1}^{D}\|x_i\|+\prod_{i=1}^{D}\|x_i\|$
F14: Schwefel's problem 2.21	$f_{14}(x)=\max_i\{\|x_i\|,1\le i\le D\}.$
F15: Generalized penalized function 1	$f_{15}(x)=\frac{\pi}{D}\left\{10\sin^2(\pi y_1)+\sum_{i=1}^{D-1}(y_i-1)^2(1+10\sin^2(\pi y_{i+1}))+(y_D-1)^2\right\}$ $+\sum_{i=1}^{D}u(x_i,10,100,4),\quad y_i=1+\frac{1}{4}(x_i+1)$
F16: Generalized penalized function 2	$f_{16}(x)=0.1\left\{\sin^2(3\pi x_1)+\sum_{i=1}^{D-1}(x_i-1)[1+\sin^2(3\pi x_{i+1})]+(x_D-1)[1+\sin^2(2\pi x_D)]\right\}$ $+\sum_{i=1}^{D}u(x_i,5,100,4)$
F17: Kowalik's function	$f_{17}(x)=\sum_{i=1}^{11}[a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}]^2$
F18: Six-hump camel-back function	$f_{18}(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4.$
F19: Branin function	$f_{19}(x)=(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6)^2+10(1-\frac{1}{8\pi})\cos x_1+10.$
F20: Hartman's function 1	$f_{20}(x)=-\sum_{i=1}^{4}c_i\exp[-\sum_{j=1}^{3}a_{ij}(x_j-p_{ij})^2].$
F21: Hartman's function 2	$f_{21}(x)=-\sum_{i=1}^{4}c_i\exp[-\sum_{j=1}^{6}a_{ij}(x_j-p_{ij})^2]$
F22, 23 and 24: Shekel's family	$f(x)=-\sum_{i=1}^{m}[(x-a_i)^T(x-a_i)+c_i]^{-1}$,with m=5,7 and 10 for F22, F23 and F24.

3. Experiments on DE/cluster

This section will present some experiments to show the performance of DE/cluster in its efficiency and the effect of some key parameters setting and the clustering component in DE/cluster. Experiment 1 is designed to compare the efficiency of DE/cluster and the traditional DE, SaDE, JADE. Experiment 2 is designed to show the ability the DE/cluster on some test functions that are non-linearly separable to demonstrate its ability of searching only based on its mutation strategy. Experiment 3 shows the runtime characteristic of the cluster information learned by DE/cluster on different kinds of test functions. Experiment 4 do a simple analysis on the running time of DE/cluster and simulations are carried out to prove that the time cost of DE/cluster is under control.

3.1 Experiments Setting

In the following tests DE/cluster is configured as follows: the population used *NP*=50, the generator of *F* is *Gaussian*(0.4,0.2). The initial value of *CR*=0.65 and *CRstd* is set as 0.1. The generations *LP* to store the memory of *CR* is set as 50. The probability to do GA crossover *Alpha*=0.25/*cc*, where cc is the number of clusters. The differential pool's size is configured as *M*1=10, *M*2=20. The maximum size of a cluster allowed *S* is set as 40% of the whole population. The frequency to do cluster analysis is set as every *K*=5 generations.

The algorithms to be compared with DE/cluster are as follows:

a) Traditional DE: DE/rand/1/bin (*F*=0.5, *CR*=0.3) and DE/rand-to-best/2 (*F*=0.5, *CR*=0.3).

b) Current most efficient DE variants: SaDE(2009) and JADE.

Test functions are shown in Table 1. (Test suite used in [6], for the searching range and optimum information please refer to [6]). The test functions which are indicated shifted means that the original best solutions are shifted to new positions which are not central. Test functions which are rotated means that the variables are firstly rotated by a rotating matrix of certain condition number.

3.2 Comparison with Current Best DE Variants

To compare the performance of different algorithms, here will let the algorithms run several times on the same test suite to test whether they can find good enough solutions within a maximum function evaluations. Table 2 shows the performance comparison on the average function evaluations (FEs) cost for these algorithms to achieve a better function value on each test function than the known best value plus 1E-5 within a predefined max number of FEs (1E+5 for lower than 30-dimensional test problems while 3E+5 for 30-dimensional test functions). Results on test functions which no algorithm can solve with 100% success rate are not shown in Table 2 (such a function like F8 is actually non-linear-separable. For such a kind of problems, DE algorithms cannot perform well. In the next part we show the results of the DE/cluster algorithm on F8 compared with the other algorithms). The above tests results are all based on the average performance of 50 tests for every test function.

From the results shown in Table 2 it can be see that DE/cluster outperformed the other algorithms on most of the test functions. The performance of DE/cluster is most significant on test function F1, F5, F6, F9, F11 and F17~F24. The DE/cluster algorithm can perform well on F1 shows that its convergence speed on simple unimodal test functions is maintained while its global searching ability on the multimodal test functions are quite well. The searching speed of DE/cluster on low dimensional functions F17-F24 is impressive.

Table 2. Performance comparison on the average function evaluations cost within a predefined max value. Best performance on each test function is presented in "bold" font.

		DE/cluster		DE/rand/1/bin (0.5,0.3)		DE/rand-to-best/2/bin (0.5,0.3)		SaDE(2009)		JADE	
D	***f***	***NFE***	***SR***	***NFE***	***SR***	***NFE***	***SR***	***NFE***	***SR***	***NFE***	***SR***
10	1	**4687**	100%	10291	100%	10058	100%	8375	100%	6852	100%
10	2	**10352**	100%	72436	100%	53658	100%	14867	100%	10930	100%
10	3	-	96%	-	0%	-	0%	**42446**	100%	-	92%
10	4	17647	100%	-	83%	71278	100%	15754	100%	**11724**	100%
10	5	**7316.1**	100%	15157	100%	15045	100%	12123	100%	10730	100%
10	6	**7700.2**	100%	16682	100%	16980	100%	12244	100%	10939	100%
10	7	-	90%	**29961**	100%	59205	100%	35393	100%	32019	100%
10	9	**14456**	100%	23155	100%	30621	100%	23799	100%	18017	100%
10	11	**15459**	100%	29559	100%	46592	100%	26945	100%	17886	100%
10	12	**11973**	100%	14698	100%	33091	100%	16663	100%	17460	100%
30	1	**14326**	100%	34687	100%	31470	100%	20184	100%	22226	100%
30	2	85450	100%	-	0%	-	0%	118743	100%	**72884**	100%
30	5	**21074**	100%	49822	100%	45948	100%	26953	100%	31450	100%
30	6	**22664**	100%	55108	100%	49961	100%	33014	100%	31286	100%
30	7	**-**	98%	**39436**	100%	41314	100%	-	80%	-	98%
30	9	**49683**	100%	-	0%	-	0%	58723	100%	1.09E+05	100%
30	11	**49231**	100%	-	0%	-	0%	77920	100%	1.09E+05	100%
30	12	46303	100%	73756	100%	-	0%	**44283**	100%	1.08E+05	100%
30	13	19792	100%	39460	100%	**18617**	100%	25137	100%	37430	100%
30	14	1.46E+05	100%	149511	100%	-	0%	**88934**	100%	1.65E+05	100%
30	15	16660	100%	32420	100%	**14289**	100%	18742	100%	21806	100%
30	16	20768	100%	31346	100%	-	93%	**19390**	100%	22236	100%
4	17	**5092.2**	100%	31121	100%	-	93%	6426	100%	-	98%
2	18	**644.02**	100%	2178	100%	1416	100%	2076	100%	894	100%
2	19	**764**	100%	2246	100%	1593	100%	2614	100%	1492.2	100%
4	20	**459.16**	100%	926	100%	1004	100%	802	100%	484.2	100%
6	21	**1158.6**	100%	3970	100%	4759	100%	3080	100%	-	76%
4	22	**2616.5**	100%	10468	100%	12381	100%	4947	100%	-	82%
4	23	**2040.1**	100%	8653	100%	12921	100%	4173	100%	-	96%
4	24	**1738.4**	100%	8742	100%	14933	100%	4267	100%	-	92%

3.3 Test the Searching Ability of DE/cluster on the Non-linearly-separable Test Function

Currently DE usually cannot perform well on some linearly inseparable multimodal functions. In this experiment we will test the performance of DE/cluster on F8 on lower dimensions to demonstrate its performance on linearly inseparable multimodal functions (with the dimension increasing the problem becomes more and more difficult to solve). For the linearly inseparable multimodal functions, the *CR* adaptation will not work and the algorithm will mainly depend on its mutation strategy. Test is carried out on F8 (with *D*=3, 4 and 5) and the success rate for different algorithms within maximum 2E+5 functions are showed in Table 3.

For all the test algorithms, *CR* is set as 1. The adaptation of *F* is retained for the algorithms which apply *F* adaptation. For the DE/rand algorithm, *F* is randomly generated by a Gaussian distribution (0.5, 0.2). From the results we can see that for all the test algorithms tested, their performance on F8 degenerates quickly with the dimension increased from 3 to 5. This is because of the exponential increasing of the number of local minima. To solve this kind of test functions, the algorithm must have good global search ability. The tested algorithms are with different kinds of global search ability enhancing strategies. For the traditional DE algorithms, DE/rand has the best global searching ability and the slowest searching speed because it uses randomly selected as the base vector. JADE applies multiple top individuals as the guiding vector but with a useful F adaptation strategy. SaDE can be taken as an algorithm with mixed strategies which its performance should be tested in this test.

The results shows that the performance on the tested functions DE/cluster > JADE > SaDE > DE/rand. The reason why DE/cluster can perform better than the other algorithms on F8 is that the local restart module applied in DE/cluster can make a better balance of searching speed and global search while the *SPEX* cluster can do a good job on maintain the population diversity. The results showed DE/rand performed the worst in opposite with the fact that it applies the most global mutation strategy, this may caused by the inefficiency of the mutation operator applied, i.e., it is global, but it may be inefficient to generate a new promising result to help the algorithm find a new minimum point.

3.4 Runtime Characteristic of the Clustering Results of DE/cluster on Different Test Functions

Here will study the cluster information DE/cluster mined in the evolution procedure. Fig. 4 shows the number of clusters found in every generation for different test functions (average results for 50 tests). Fig. 4 shows the clusters count for F1 during the evolution procedure. It can be seen that the cluster count shows clearly decreasing trend with certain concussion. This shows the DE/cluster learned from the distribution the population and adapted it into the searching mode of a unimodal function (with less clusters, the algorithm will searching faster). The trend line of the clusters count for F7 (Fig. 5) is quite different with that of F1. In the first stage the trend line decreasing quickly while in the second stage the trend line keeps climbing. This is actually because of the characteristic of F7: at the beginning searching stage of F7 with the large beginning searching range, the function is almost like a unimodal function while when the searching range is small, many local minima emerged and the function becomes to a multimodal function. The trend line of F7 shows DE/cluster can adapt its searching mode between unimodal and multimodal to a certain degree.

The performance of DE/cluster on F14 is not as good as the other DE algorithms as shown above. The reason may be explained by Fig. 6. It shows that the clusters count of F14 is in great vibrating from the beginning to the end of the evolution procedure. This shows the DE/cluster algorithm is confused by the distribution of the population and cannot find an efficient mode for the test function. The reason of this phenomenon should be further studied

Table 3. Comparison on linear inseparable function F8 (D=3, 4 and 5)

	Success rate within 2E+5 NFEs within 50 tests			
	DE/cluster	***JADE***	***SaDE***	***DE/rand***
F8 (d=3)	**100%**	**100%**	78%	68%
F8 (d=4)	**78%**	62%	16%	6%
F8 (d=5)	**24%**	2%	0%	0%

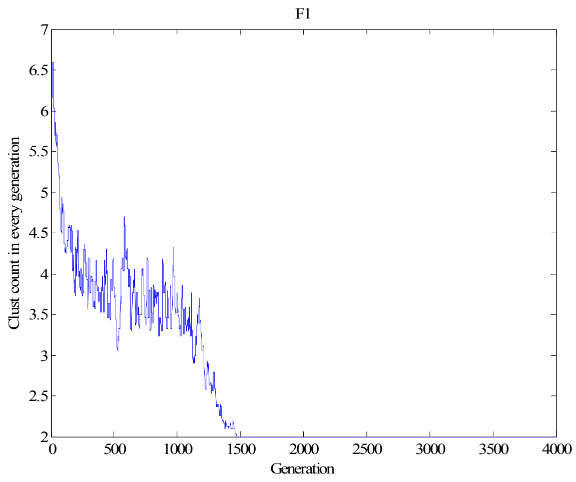

Figure 4. Clusters count during evolution for F1.

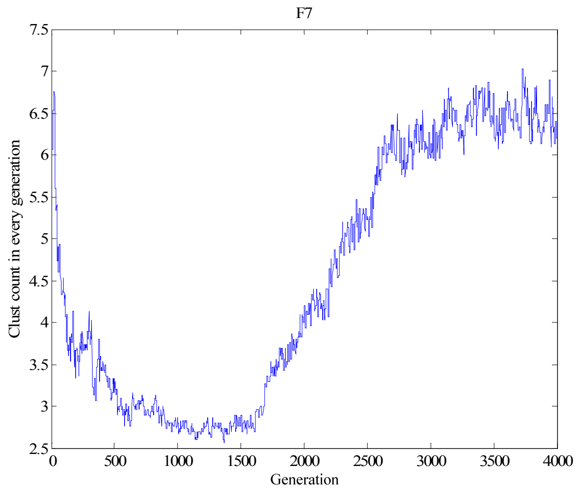

Figure 5. Clusters count during evolution for F7.

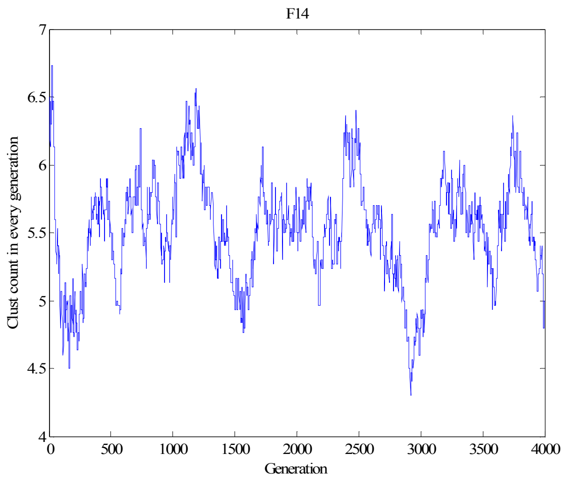

Figure 6. Clusters count during evolution for F14

3.5 Run time analysis of DE/cluster

The time cost of DE/cluster can be controlled greatly due to the fact that the clustering of the population is changing slowly during the evolution procedure. The usually changing pattern of the clusters of the population is often suddenly-changing, i.e., with some new promising individuals found, some individuals are attracted and form a new cluster. For a single clustering process, its time complexity is caused by computing the distance matrix and sorts the matrix to some extent. The computation of the distance matrix, the time complexity is about $O(D*NP^2)$, which is because the distance between every pair of individuals in the population is needed. For the sorting part of clustering, at the worst case when a total sort of the distance matrix is done, the time complexity is $O(NP^2*\log NP)$. Usually the number *NP* of individuals used for DE is set as *D**10, so the second part of the time cost is actually smaller than the first part. Thus theoretically the time cost of the clustering process in DE/cluster is mainly generated by computing the distance matrix. Thus we concluded that it is acceptable to do the explicit clustering analysis after the distance matrix is already computed on time consuming aspect. As has been mentioned above, the time consuming of DE/cluster can be greatly improved by doing cluster analysis only when it is needed. Table 4 shows the time cost comparison of DE/cluster with other algorithms. For DE/cluster, the number *K* of generations to do re-clustering is set as 30. Form the test results shown in Table 4 we can see that it is acceptable (compared with the DE/rand algorithm and SADE). The time consuming of doing cluster analysis can be further neglected when function evaluation is time costly.

Table 4. Run time comparison

	Time cost (s)				
DE variants	**F9 (d=10)**	**F9 (d=30)**	**F1 (d=30)**	**F1 (d=10)**	**F21**
DE/ cluster	0.4168	2.3075	0.6828	0.1682	0.0487
DE/ rand/1	0.3137	~	0.3634	0.0977	0.0666
SADE	0.5529	2.3721	0.6619	0.1765	0.0637

4. Conclusion

In this paper we have proposed a new DE algorithm called DE/cluster. DE/cluster applies the simple hierarchical clustering algorithm to mine the distribution information of the population and use this information to partition the population dynamically. The cluster analysis will be done for every some generations or when the local restart has happened, so the time cost of cluster analysis can be controlled. Clustering can make the algorithm be able to adapt the partition at different searching phase and can make the subpopulations searching for their local minima more efficiently. DE/cluster applies a pool to provide individuals for the subpopulations which have not enough individuals to generate differential vectors. Different mutation strategies are chosen for different kinds of clusters. The special cluster combined from all the single-individual-clusters is used for global search with the DE/current-to-best strategy and for the other clusters more greedy mutation strategy DE/best are applied, thus different searching strategies are used for different individuals in the current population according to their distribution characteristics, which is not the same as SaDE.

The local restart strategy applied in DE/cluster can make the subpopulations which have converged to local minima restart and searching the other space. This makes DE/cluster even perform better than DE/rand on some non-linear-separable test functions.

5. ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China (NSFC) No. 61070004, and by NSFC Joint Fund with Guangdong under Key Project U0835002. The authors are with the Key Lab of Digital Life, MoE, China, and the Key Lab. of Softw. Tech, Edu. Depart of Guangdong Province. The corresponding author is Jun Zhang, email: junzhang@ieee.org

6. References

[1] R. Storn and K. V. Price. *Differential evolution–A simple and efficient adaptive scheme for global optimization over continuous spaces*. Tech. Report TR-95-012, Institute of Company Secretaries of India, Chennai, Tamil Nadu, 1995.

[2] R. Storn and K. V. Price. Differential Evolution–a simple and efficient heuristic for global optimization over continuous spaces. *J. Global Optimization*, 11, 4 (1997), 341–359.

[3] J. H. Holland. *Adaptation Natural and Artificial Syst*. Ann Arbor, MI: Univ. Michigan Press, 1975.

[4] J. Kennedy and R. Eberhart. Particle swarm optimization. in *Proc. IEEE Int. Conf. Neural Netw*., 1995, 1942–1948.

[5] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE Trans. Evol. Comput.*, 10, 6 (Dec. 2006), 646–657.

[6] A. K. Qin, V. L. Huang, and P. N. Suganthan. Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Trans. Evol. Comput.*, 13, 2 (Apr. 2009), 398–417.

[7] Swagatam Das, Ajith Abraham, and Amit Konar. Differential Evolution Using a Neighborhood-Based Mutation Operator. *IEEE Trans. Evol. Comput.*, 13, 3 (Jun. 2009), 526–552.

[8] Jingqiao Zhang, and Arthur C. Sanderson. JADE: Adaptive Differential Evolution with Optional External Archive. *IEEE Trans. Evol. Comput.* ,13, 5 (Jun. 2009), 945–958.