2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Variable Grouping Based Differential Evolution Using An Auxiliary Function for Large Scale Global Optimization

Fei Wei
School of Computer Science and Technology
Xidian University, Xi'an, China
Email: feiweixjf@gmail.com

Yuping Wang
School of Computer Science and Technology
Xidian University, Xi'an, China
Email: ywang@xidian.edu.cn

Tingting Zong
School of Computer Science and Technology
Xidian University, Xi'an, China
Email: TingtZong@163.com

***Abstract*—Evolutionary algorithms (EAs) are a kind of efficient and effective algorithms for global optimization problems. However, their efficiency and effectiveness will be greatly reduced for large scale problems. To handle this issue, a variable grouping strategy is first designed, in which the variables with the interaction each other are classified into one group, while the variables without interaction are classified into different groups. Then, evolution can be conducted in these groups separately. In this way, a large scale problem can be decomposed into several small scale problems and this makes the problem solving much easier. Furthermore, an auxiliary function, which can help algorithm to escape from the current local optimal solution and find a better one, is designed and integrated into EA. Based on these, a variable grouping based differential evolution algorithm (briefly, VGDE) using auxiliary function is proposed. At last, the simulations are made on the standard benchmark suite in CEC'2013, and VGDE is compared with several well performed algorithms. The results indicate the proposed algorithm VGDE is more efficient and effective.**

I. Introduction

Many real-world optimization problems involve a large number of decision variables. How to handle this sort of real-world large scale global optimization (LSGO) problems efficiently still remains an open problem. In recent years, many works have been reported for solving LSGO problems. In these existing approaches, cooperative coevolution (briefly, CC) [1]- [8], decomposition method [9] and estimation of distribution algorithm [10], are the typical examples of the most efficient and popular algorithms, where CC, proposed by Potter and De Jong [1], adopted a divide-and-conquer strategy and was especially attractive. In the early stage of the development of CC, one-dimension based and splitting-in-half methods were adopted, while the interaction between variables had not been considered in these methods. Thus, they could not solve problems consisting of non-trivial variable interactions efficiently. In order to mitigate this problem, recently, some effective grouping methods (e.g., random grouping method [3], [4], variable interaction learning grouping strategy [5]) were successively proposed. However, random grouping method just increased the probability of two interacting variables being allocated into the same group. Variable interaction learning grouping strategy [5] considered the variable correlation, and thus it performs better than random grouping method; however, the condition used to verify the variable correlation is only a sufficient condition (not a sufficient and necessary condition), thus the variables with interactions in some cases can not be correctly classified into one group, in other word, this strategy will incorrectly group some interacting variables in some cases.

To overcome this shortcoming, in this paper, we propose a novel variable grouping strategy that can more accurately and efficiently group the variables according to their interaction.

Except for the novel grouping strategy, in order to further improve the efficiency and effectiveness of the proposed algorithm, an auxiliary function is also designed and combined into the algorithm design. The auxiliary function can help the algorithm to escape from the current local minimal solution, and find a better local optimal solution.

Based on the above ideas, a variable grouping based differential evolution algorithm (VGDE) is proposed, and VGDE has the following advantages: 1) can decompose a large scale problem to several small scale problems if variables are separable; 2) can search for multiple areas in the search space simultaneously (search for each decomposed problem respectively); 3) can escape from the current best local optima easily via the auxiliary function.

The simulations are made on 15 problems in CEC'2013 benchmark suite [12], and VGDE is compared with several efficient algorithms. The results indicate that VGDE is more efficient.

The reminder of the paper is organized as follows. In section II, a variable grouping strategy is proposed. In section III, an auxiliary function is introduced. Section IV presents a novel algorithm framework VGDE. Numerical experiments are given in Section V. Section VI presents conclusions and future works.

In this paper, we adopt the following notations:
k : the generation number;
x_k^*: the local minimizer of the objective function in the k-th generation;
f_k^*: the function value at x_k^*;
x^*: the global minimizer of the objective function;
$MaxFEs$: the maximum number of function evaluations.

II. VARIABLE GROUPING STRATEGY

Decomposition of the original problem into several smaller subcomponents is a critical step of the proposed algorithm. The ideal goal of decomposition is that those decision variables with interaction are classified into one group, and those decision variables without interactions are classified into different groups. At present, some grouping strategies have been proposed, such as random grouping (DECC-G [3], MLCC [4]), variable interaction learning grouping (CCVIL [5]), route distance grouping [6], differential grouping [7], delta grouping [8], and so on. However, these grouping strategies still cannot realize accurate grouping for some problems. For example, random grouping just increases the probability of two interacting variables being allocated into the same group.

To overcome this shortcoming, in this paper, we propose a new adaptive variable grouping method (briefly, AVG). The AVG framework is as follows: first, a set $N\text{-}Sep$ is built whose elements are the factors determining variables interaction; second, matching these elements to the test problem; finally, if an element is matched, putting the variables connected by the element into a group.

In the following, we explain how to build the set $N\text{-}Sep$. Note that a general test function consists of a finite number of four arithmetic operations "+, -, $\times$ and $\div$" and composite operation of basic elementary functions (i.e., power function y^a, exponential functions a^y and e^y, logarithmic functions $\ln y$ and $\log_a y$, trigonometric functions $\sin y$, $\cos y$, $\tan y$, $\cot y$, $\sec y$ and $\csc y$, inverse trigonometric functions $\arcsin y$, $\arccos y$, $\arctan y$, $\operatorname{arccot} y$, $\operatorname{arcsec} y$ and $\operatorname{arccsc} y$ and constant, where $y \in R$). We build the set $N\text{-}Sep$ based on this formulation characteristic of a general function in the following cases:

(1) Variables separability in four arithmetic operations. If function $p_1(x) = a_1x_1 + a_2x_2 + \cdots + a_mx_m$, then each a_ix_i in this function can be optimized independently and thus the variables $x_1, x_2, \cdots, x_m$ in this function are separable, where $a_i \in R$, $i = 1, 2, \cdots, m$. While if a function contains "$\times$" or "$\div$" of two variables, these two variables can not be optimized independently and thus they are non-separable. We put "$\times$" and "$\div$" into a set $N\text{-}Sep$.

(2) Variables separability in a composite function. For a basic elementary function $g(y)$ with $y \in R$ and an n-dimensional function $h(x)$, if $g(y)$ is monotone and variables in function $h(x)$ are separable, then variables in composite function $g(h(x))$ are also separable (e.g., if $g(y) = e^y$, and $h(x) = x_1 + x_2 + \cdots + x_{10}$, then variables in $g(h(x)) = e^{x_1+x_2+\cdots+x_{10}}$ are separable). Otherwise, variables in $g(h(x))$ are non-separable. We put the non-monotonic functions (e.g., trigonometric functions, inverse trigonometric functions, and power function $g(y)^a$ with $y \in R$, where a is an even number) into $N\text{-}Sep$.

(3) Variables separability in a function obtained by one operation of "+", "−", "×" and "÷" on two composite functions in (2). The variables in "×" or "÷" of two composite functions in (2) are non-separable except that both composite functions are exponential functions. We put "×" and "÷" into a set $N\text{-}Sep$ (except that both composite functions are exponential functions).

These three cases can construct set $N\text{-}Sep$, and the variables can be classified into different groups by the following algorithm 1.

Algorithm 1: The flow of the adaptive variable grouping strategy AVG

1 The elements in $N\text{-}Sep$ can be seen as strings (such as "cos" and "sin");
2 Using regular expressions to match these strings in $N\text{-}Sep$ for each problem in benchmark suite;
3 If a string is matched from the problem, then these variables contained in this string will be classified into a group, and so on;
4 If some variables are not matched, then they are separable, and each variable is put into a group.

III. AN AUXILIARY FUNCTION

It is often occurred that the number of local optimal solutions increases exponentially with respect to the dimensions for many problems. This makes EAs be easy to be trapped into local optimal solutions and makes global optimization solving become a great challenge. Thus, one of the key issues for the global optimization problem is effectively handling a large number of local optimal solutions and finding the global optimal solution as quickly as possible. To achieve this purpose, we use an auxiliary function to help the proposed algorithm to jump out the current local optimal solution and arrive at a better one. An auxiliary function is used as follows (for more details, please refer to literature [11]):

$$P(x, x_k^*) = -\|x - x_k^*\|^2 g(f(x) - f(x_k^*)), \tag{1}$$

$$g(t) = \begin{cases} \pi/2, & t \geq 0, \\ r \cdot \arctan(t^2) + \pi/2, & t < 0. \end{cases}$$

where r is an adjustable positive real number large enough, used as the weight factor.

The auxiliary function $P(x, x_k^*)$ has the following properties.

Theorem 1: Suppose that x_k^* is a local minimizer of $f(x)$, then x_k^* is a strictly local maximizer of $P(x, x_k^*)$; and for any $x \in \Omega_1 = \{x | f(x) \geq f(x_k^*), x \in \Omega, x \neq x_k^*\}$, $\nabla P(x, x_k^*) \neq 0$.

This theorem illustrates that any solution x, which is no better than the current best solution x_k^*, can not be a local optimal solution of auxiliary function.

Theorem 2: Suppose that x_k^* is a local minimizer of $f(x)$, and $\Omega_2 = \{x | f(x) < f(x_k^*), x \in \Omega\}$ is not empty, then there exists a point $x_k' \in \Omega_2$, such that x_k' is a local minimizer of $P(x, x_k^*)$.

This theorem illustrates that a better local optimal solution than the current best solution of the original function can be obtained by minizing the auxiliary function. Thus, we can jump out the current best local optimal solution via minimizing the auxiliary function at x_k^*.

The key issue for constructing the auxiliary function is to know a local optimal solution x_k^*. How to get a local optimal solution x_k^*? When a good solution x_k is found in the evolution process, it may not be a local optimal solution.

However, a local optimal solution x_k^* can be obtained by a local search algorithm using x_k as an initial solution. There are many existing local search algorithms (e.g., Conjugate Gradient Method, Newton Method and Quasi Newton Method, etc.), but they require the gradients of the function, therefore, these methods are not suitable for solving non-differentiable problems. To be applicable to the non-differentiable problems and avoid computing the gradients, a revised version of Quasi Newton algorithm is designed as follows.

Algorithm 2: The pseudocode of the local search strategy

Initialization: Choose a tolerance $\varepsilon > 0$, e.g. $\varepsilon = 1.0e-5$, and a small constant $\delta \in (0,1)$, $\sigma \in (0, 0.5)$, and $\Delta x = 1.0e-1$.
Give an initial point x_0 and an approximate inverse of the Hessian matrix B_0,
$k = 0$.
Calculate $g_k = (g_k^1, g_k^2, \cdots, g_k^n)^T$, where $g_k^i = (f(x_k + \Delta x e_i) - f(x_k))/\Delta x$ and $e_i = (0, 0, \cdots, 1, 0, \cdots, 0)^T$ with 1 being the i-th component of e_i for $i = 1 \sim n$.
repeat
  Obtain a direction d_k by solving $B_k d_k = -g_k$.
  Perform a line search based on the Armijo criterion [16] to find an acceptable stepsize $\lambda_k = \delta^{m_k}$, where m_k is the smallest non-negative integer that satisfy the following inequality:

$$f(x_k + \delta^{m_k} d_k) \leq f(x_k) + \sigma \delta^{m_k} g_k^T d_k.$$

  Set $s_k = \lambda_k d_k$, $x_{k+1} = x_k + \lambda_k d_k$ and $k = k + 1$, then calculate g_k according to Step 2 and set $y_{k-1} = g_k - g_{k-1}$, then

$$\begin{cases} B_k = B_{k-1} + (\beta_{k-1} s_{k-1} s_{k-1}^T - B_{k-1} y_{k-1} s_{k-1}^T \\ -s_{k-1} y_{k-1}^T B_{k-1})/s_{k-1}^T y_{k-1}, \\ \beta_{k-1} = 1 + y_{k-1}^T B_{k-1} y_{k-1}/s_{k-1}^T y_{k-1}. \end{cases}$$

until $\|g_k\| \leq \varepsilon$;
$x^* = x_k$,
return x^*.

After a local optimal solution can be obtained by algorithm 2, in order to jump out this local optimal solution and go into a deeper valley, we can minimize the auxiliary function as shown in theorem 2. A pseudocode of minimizing the auxiliary function is shown in algorithm 3.

The auxiliary function minimization algorithm can only escape from the current local minimizer and reach to a lower local minimizer, however, it cannot search multiple regions simultaneously. To search multiple regions simultaneously, we combine it with evolutionary algorithm and design a novel algorithm in the next section.

IV. Variable Grouping Based Differential Evolution using auxiliary function (VGDE)

As mentioned before, a large scale optimization problem can be decomposed into several small scale optimization problems by algorithm1. In this paper, we use self-adaptive differential evolution with neighborhood Search(SaNSDE) [15] to optimize each sub-problem (corresponding to each group) separately. It has been indicated that SaNSDE performs significantly better than other similar algorithms due to its self-adapted crossover rate CR and scaling factor F [15], and SaNSDE has been successfully applied in a variety of problems [3], [4]. Furthermore, in order to enhance this algorithm, we integrate an auxiliary function into it. The pseudocode is given

Algorithm 3: The pseudocode of minimizing the auxiliary function algorithm

Input: x_0 is an initial point, and ε is a small constant
repeat
  Minimize object function $f(x)$ from x_0 by using algorithm 2 to obtain a local minimizer $x^{'}$;
  Construct an auxiliary function $P(x)$ at $x^{'}$;
  Minimize $P(x)$ from $x^{'}$ by using algorithm 2 to obtain a local minimizer x^*;
  Calculate $\sigma = f(x_0) - f(x^*)$;
  if $\sigma > \varepsilon$ **then**
    $x_0 = x^*$;
  end
until $\sigma \leq \varepsilon$;
return x^*.

Algorithm 4: Pseudocode of VGDE

Initialization: let $K = 0$, and $FEs = 0$. Choose two positive real numbers λ, r large enough, $\sigma = \infty$, $f(x_0^*) = f(x_0^{'}) = \infty$, ε is a tolerance threshold.
Perform the algorithm 1, and obtained M groups;
repeat
  $\sigma = \infty$;
  generate initial population POP via uniform distribution, where the population size is NP;
  while $\sigma > \varepsilon$ **do**
    for $j \in [1, \cdots, M]$ **do**
      $k = 1$;
      perform algorithm SaNSDE on each group;
      output the iterations $k1$ in SaNSDE, the current best solution $x_k^{'}$ and second-best solution $x_{k-1}^{'}$, and the number of function evaluation $SFEs$ in SaNSDE, then update FEs=FEs+SFEs, and $\delta = f(x_{k-1}^{'}) - f(x_k^{'})$;
      if $\delta \leq \varepsilon$ **then**
        perform algorithm 3 on the auxiliary function at $x_k^{'}$, then obtain a current best solution x_k^*; update $\delta = f(x_k^{'}) - f(x_k^*)$;
      else
        $x_k^* = x_k^{'}$;
      end
    end
    update current best solution $x_K^* = x_k^*$, and $\sigma = f(x_K^*) - f(x_{(K-1)}^*)$;
    $K = K + 1$;
  end
until *the maximum number of function evaluations MaxFEs is met*;
$x^* = x_K^*$, and $f(x^*) = f(x_K^*)$;
return x^* and $f(x^*)$.

in Algorithm 4.

V. Numerical experiments

A. Benchmark suite and parameters setting for VGDE

- In this section, the proposed algorithm VGDE is tested on CEC'2013 benchmark suite [12].
- In experiments, VGDE was tested on an Intel(R) Core(TM) i7 CPU 870 with 2.93GHz in Matlab R2012a.
- Population size: $N = 50$.
- Parameters in algorithm VGDE: $\lambda = 100$, $r = 100$, $\varepsilon = 1.0e-7$.

B. The simulation results

To verify the efficiency of the proposed variable grouping strategy and the auxiliary function, we replace the proposed variable grouping strategy in VGDE by the randomly grouping strategy [4] and deleting the auxiliary function, and the resulted algorithm is denoted by RVGDE.

VGDE and RVGDE are executed 25 independent runs for each test problem, respectively, and we record the results of the best, the worst, the mean, and the median solutions as well as the standard deviation in Table I with $MaxFEs = 3.0e6$. Table I also lists these results of three other compared algorithms (SACC [17], MOS [18] and DECC-G [3]) on CEC'2013 benchmark suite. The only difference between VGDE and SACC [17] is that the grouping strategy is different. The algorithm MOS in [18] is one of the best algorithms in CEC'2013 competition. DECC-G [3] is a representative algorithms in large scale global optimization.

In Table I, the function values at the best solution, the worst solution, the median solution obtained in 25 runs are denoted as "Best", "Worst" and "Median", respectively. Also, The mean value and the standard deviation of these function values are denoted as "Mean" and "Std", respectively. A two-tailed t-test was only conducted between VGDE and MOS, since VGDE clearly outperformed RVGDE, SACC and DECC-G.

Note that the bold form of the results in Table I means that the best result obtained by all algorithms listed. From Table I, we can see that the all performance index values of $f1$, $f2$ and $f3$ obtained by VGDE are the best, and the mean values obtained by VGDE on seven problems are better than those obtained by the other four algorithms. which indicates that the proposed VGDE are more efficient. Also, the mean values obtained by VGDE on all problems are better than those obtained by RVGDE, which indicates that the proposed variable grouping strategy (AVG) and auxiliary function are effective. The mean values of VGDE are better than those of SACC, which also indicates that AVG in VGDE is very effective. Moreover, the mean values of seven problems in VGDE are better than those of MOS, and the mean values of thirteen problems in VGDE are better than those in DECC-G, which indicate that AVG and auxiliary function in VGDE are effective.

From another point of view, first, for fully separable functions $f1$, $f2$ and $f3$, the results show that VGDE is more effective than other algorithms.This indicate the proposed AVG is more effective. Second, for eight partially additively separable problems $f4$-$f11$, almost all of the mean values in VGDE are better than those in RVGDE and SACC; the mean values of five problems $f5$, $f6$, $f7$, $f9$, $f10$ in VGDE are better than those in MOS; the mean values of six problems $f4$, $f5$, $f7$, $f9$, $f10$, $f11$ in VGDE are better than those in DECC-G. Overall, VGDE is more effective to partially additively separable functions. Third, for overlapping functions $f12$-$f14$ and nonseparable function $f15$, the results of $f12$-$f15$ in VGDE are better than those in the other three algorithms(RVGDE, SACC, DECC-G). It indicates that auxiliary function in VGDE is effective for overlapping and nonseparable functions. In addition, one of the main difference between SACC and DECC-G is that SACC contain an auxiliary function, from the table 1, we can seen that the results in SACC are better than those in DECC-G, so the auxiliary function is effective. However, the results of $f12$-$f15$ in VGDE are worse than those in MOS, this might be because the evolution algorithm used in the proposed algorithm is easily trapped in the local optimum and appeared premature convergence.

From the results of two tailed T-test, it can be seen that the results between VGDE and MOS are nearly identical for two problems $f1$ and $f6$; and the six problems' results of VGDE are obviously better than those of MOS. However, for overlapping functions and nonseparable function, the results of VGDE are worse than MOS, and this may be due to the prematurity of evolutionary algorithm used, and this will be a problem to be solved in the future.

In order to show the advantage of the proposed algorithm VGDE more intuitively, we use the semilog line diagram to plot the convergence curve of problems. Fig.1 to Fig.3 show that the semilog line diagram of the three selected functions: $f1$, $f7$, and $f11$. Plotted by FEs on the horizontal axis and the function value of each problem on the vertical one, where the vertical one is logarithmic scale. For each selected function, the convergence curves of three algorithms are plotted using the average results over all 25 runs. The thick line represents the convergence curve of VGDE, the thin line represents the convergence curve of SACC, and the dotted line represents the convergence curve of RVGDE.

From Fig.1 to Fig.3, it can be seen that VGDE is much better than two other algorithms: at the early stages of the evolution, the decreasing degree of function value by VGDE is larger than that of function value by other algorithms. The reason may be that AVG in VGDE is very effective; at the later evolution process, VGDE also has the very good results compared with the other two algorithms, the reason may be that the auxiliary function is effective.

However, at the later evolution process, the evolution become slower and the global optimal solutions have not been found, such as in Fig.3. One of the reasons is that the population may be premature. In addition, because of the difficulty of problems and the inappropriate parameters chosen, the variable grouping strategy and the auxiliary function is likely to become ineffective.

Overall, both the data in Table I and Figures 1-3 show that the proposed algorithm VGDE is more effective and efficient.

Nevertheless, the results for partially additively separable

functions $f4 - f11$ and overlapping nonseparable functions $f12 - f15$ are far from their real global optimal values. This may be caused by several reasons. Firstly, the parameters are difficult to choose and they may be not appropriately chosen. Secondly, for overlapping function and nonseparable function, the grouping strategy AVG in VGDE has a little effect on improving the performance of the algorithm. Finally, the chosen number of function evaluations is not enough to get a satisfied solution.

VI. Conclusions

In this paper, a variable grouping based differential evolution with an auxiliary function(briefly, VGDE) is proposed. The variable grouping strategy can divide the interacting variables into a group and to achieve the goal of becoming large scale problems into several small scale problems, and the auxiliary function can help the proposed algorithm to jump out the current local optimal solution and reach to another better local optimal solution. VGDE can search multiple regions simultaneously and thus has more possibility to find a better local optimal solution. The experiment results also indicate that VGDE is efficient for large scale problems and more efficient than the compared algorithms.

There are several relevant issues to be addressed further in the future. Firstly, a variable grouping method for large scale global optimization needs to be further explored. Secondly, self-adaptive mechanism for parameters in the various evolution operators is also very worthy to study. Thirdly, it is necessary to revise VGDE to be more efficient to the large scale optimization problems.

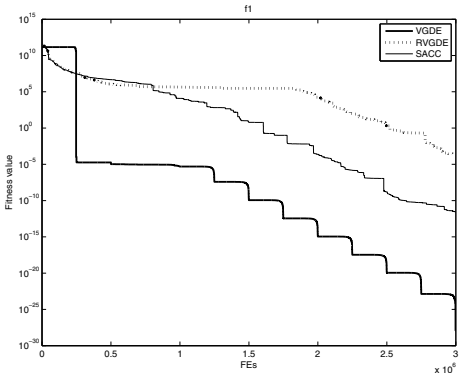

Fig. 1. The convergence curve of three algorithms on f1

TABLE I. Comparison between VGDE and other algorithms on 1000-D functions

P		VGDE	RVGDE	SACC	MOS	DECC-G	p-value (VGDE-MOS)
f1	Best	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	1.75e-13	NaN
	Median	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	2.00e-13	
	Worst	**0.00e+00**	1.81e-03	6.81e-23	**0.00e+00**	2.45e-13	
	Mean	**0.00e+00**	2.58e-04	2.73e-24	**0.00e+00**	2.03e-13	
	Std	**0.00e+00**	6.84e-04	1.36e-23	**0.00e+00**	1.78e-14	
f2	Best	**5.97e+00**	1.91e+02	2.88e+02	7.40e+02	9.90e+02	1.18e-30
	Median	**3.78e+01**	1.15e+03	5.71e+02	8.36e+02	1.03e+03	
	Worst	**1.55e+02**	4.11e+03	2.72e+03	9.28e+02	1.07e+03	
	Mean	**4.56e+01**	2.11e+03	7.06e+02	8.32e+02	1.03e+03	
	Std	3.25e+01	1.73e+03	4.72e+02	4.48e+01	**2.26e+01**	
f3	Best	3.69e-13	**2.06e-13**	9.24e-14	8.20e-13	2.63e-10	3.07e-20
	Median	3.94e-13	**2.70e-13**	1.21e+00	9.10e-13	2.85e-10	
	Worst	**4.30e-13**	3.32e+00	3.76e+00	1.00e-12	3.16e-10	
	Mean	**3.98e-13**	3.76e-01	1.11e+00	9.17e-13	2.87e-10	
	Std	**1.42e-14**	9.96e-01	1.11e+00	5.12e-14	1.38e-11	
f4	Best	**6.07e+07**	7.02e+09	8.48e+09	1.10e+08	7.58e+09	7.21e-06
	Median	4.05e+08	2.79e+10	3.66e+10	**1.56e+08**	2.12e+10	
	Worst	2.18e+09	1.35e+11	1.71e+11	**5.22e+08**	6.99e+10	
	Mean	5.96e+08	6.09e+10	4.56e+10	**1.74e+08**	2.60e+10	
	Std	**4.45e+08**	5.46e+10	3.60e+10	7.87e+08	1.47e+10	
f5	Best	2.07e+06	**1.46e+06**	3.36e+06	5.25e+06	7.28e+14	3.97e-17
	Median	**3.05e+06**	1.02e+07	6.95e+06	6.79e+06	7.28e+14	
	Worst	**4.19e+06**	1.37e+07	1.40e+07	8.56e+06	7.28e+14	
	Mean	**3.00e+06**	8.31e+06	7.74e+06	6.94e+06	7.28e+14	
	Std	5.29e+05	4.90e+06	3.22e+06	8.85e+05	**1.51e+05**	
f6	Best	1.06e+04	1.12e+05	1.57e+05	1.95e+01	**6.96e-08**	4.16e-01
	Median	1.29e+05	1.38e+05	2.07e+05	1.39e+05	**6.08e+04**	
	Worst	1.59e+05	1.82e+05	6.00e+05	2.31e+05	**1.10e+05**	
	Mean	1.31e+05	1.46e+05	2.47e+05	1.48e+05	**4.85e+04**	
	Std	**1.74e+04**	2.56e+04	1.02e+05	6.43e+04	3.98e+04	
f7	Best	**2.27e+01**	2.44e+07	1.72e+06	3.49e+03	1.96e+08	3.52e-09
	Median	**6.42e+02**	3.13e+08	1.58e+07	1.62e+04	4.27e+08	
	Worst	**1.52e+04**	1.02e+09	1.18e+09	3.73e+04	1.78e+09	
	Mean	**1.85e+03**	4.65e+08	8.98e+07	1.62e+04	6.07e+08	
	Std	**3.39e+03**	4.22e+08	2.48e+08	9.10e+03	4.09e+08	
f8	Best	2.95e+14	5.33e+13	1.47e+14	**3.26e+12**	1.43e+14	4.32e-16
	Median	5.89e+14	1.57e+15	9.86e+14	**8.08e+12**	3.88e+14	
	Worst	1.80e+15	5.62e+15	3.08e+15	**1.32e+13**	7.75e+14	
	Mean	7.00e+14	2.14e+15	1.20e+15	**8.00e+12**	4.26e+14	
	Std	3.29e+14	1.77e+15	7.63e+14	**3.07e+12**	1.53e+14	
f9	Best	**1.44e+08**	2.54e+08	2.29e+08	2.63e+08	2.20e+08	1.50e-09
	Median	**2.33e+08**	4.04e+08	5.77e+08	3.87e+08	4.17e+08	
	Worst	**3.08e+08**	4.89e+08	1.01e+09	5.42e+08	6.55e+08	
	Mean	**2.31e+08**	3.75e+08	5.98e+08	3.83e+08	4.27e+08	
	Std	**4.01e+07**	7.97e+07	2.03e+08	6.29e+07	9.89e+07	
f10	Best	**1.17e+02**	5.92e+06	1.38e+07	5.92e+02	9.29e+04	1.26e-11
	Median	**1.51e+02**	1.09e+07	2.11e+07	1.18e+06	1.19e+07	
	Worst	**2.26e+02**	1.59e+07	7.75e+07	1.23e+06	1.73e+07	
	Mean	**1.57e+02**	1.02e+07	2.95e+07	9.02e+05	1.10e+07	
	Std	**2.51e+01**	3.16e+06	1.93e+07	5.07e+05	4.00e+06	
f11	Best	3.89e+07	3.35e+08	8.12e+07	**2.06e+07**	4.68e+10	1.82e-04
	Median	7.26e+07	1.88e+10	5.30e+08	**4.48e+07**	1.60e+11	
	Worst	1.14e+08	2.93e+11	2.30e+10	**9.50e+07**	7.16e+11	
	Mean	7.52e+07	1.01e+11	2.78e+09	**5.22e+07**	2.46e+11	
	Std	2.16e+07	1.28e+11	5.90e+09	**2.05e+07**	2.03e+11	
f12	Best	2.09e+03	2.48e+03	2.43e+02	**2.22e-01**	9.80e+02	4.97e-34
	Median	2.51e+03	2.71e+03	8.74e+02	**2.46e+02**	1.03e+03	
	Worst	3.37e+03	3.00e+03	1.72e+03	**1.17e+03**	1.20e+03	
	Mean	2.52e+03	2.71e+03	8.73e+02	**2.47e+02**	1.04e+03	
	Std	2.81e+02	1.81e+02	3.71e+02	2.54e+02	**5.76e+01**	
f13	Best	5.15e+08	3.86e+09	6.72e+08	**1.52e+06**	2.09e+10	1.05e-14
	Median	1.29e+09	4.78e+09	1.51e+09	**3.30e+06**	3.36e+10	
	Worst	3.78e+09	7.45e+09	3.40e+09	**6.16e+06**	4.64e+10	
	Mean	1.36e+09	5.21e+09	1.78e+09	**3.40e+06**	3.42e+10	
	Std	7.01e+08	1.30e+09	8.05e+08	**1.06e+06**	6.41e+09	
f14	Best	2.22e+09	3.91e+08	8.21e+07	**1.54e+07**	1.91e+11	2.20e-08
	Median	1.66e+10	7.96e+09	7.34e+09	**2.42e+07**	6.27e+11	
	Worst	6.87e+10	2.52e+11	1.10e+11	**4.46e+07**	1.04e+12	
	Mean	2.29e+10	4.75e+10	1.75e+10	**2.56e+07**	6.08e+11	
	Std	1.91e+10	9.21e+10	2.87e+10	**7.94e+06**	2.06e+11	
f15	Best	2.92e+06	3.91e+08	**1.26e+06**	2.03e+06	4.63e+07	4.99e-16
	Median	3.49e+06	6.22e+06	**1.88e+06**	2.38e+06	6.01e+07	
	Worst	3.92e+06	7.56e+06	4.90e+06	**2.88e+06**	7.15e+07	
	Mean	3.44e+06	5.32e+06	**2.01e+06**	2.35e+06	6.05e+07	
	Std	2.43e+05	2.06e+06	7.23e+05	**1.94e+05**	6.45e+06	

Acknowledgment

This work is supported by the National Natural Science Foundation of China (No. 61272119).

References

[1] Mitchell A. Potter and Kenneth A. De Jong, "A cooperative coevolutionary approach to function optimization," *in: Proceedings of International Conference on Parallel Problem Solving from Nature*, pp. 249-257, 1994.

[2] X. Li and X. Yao, "Cooperatively coevolving particle swarms for large scale optimization," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 2, pp. 210-224, Apr. 2012.

[3] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Information Sciences*, vol. 178, no. 15, pp. 2986-2999, Aug. 2008.

[4] Z. Yang, K. Tang, and X. Yao, "Multilevel cooperative coevolution for large scale optimization," *in: Proceedings of IEEE Congress on Evolutionary Computation*, pp. 1663-1670, 2008.

[5] Wenxiang Chen, Thomas Weise, Zhenyu Yang, Ke Tang, "Large-Scale Global Optimization Using Cooperative Coevolution with Variable Interaction Learning," *PPSN*, vol. 6239, pp. 300-309, 2010.

[6] Mei, Y., Li, X., Yao, X., "Cooperative Co-evolution with Route Distance Grouping for Large-Scale Capacitated Arc Routing Problems," *IEEE Transactions on Evolutionary Computation*, (accepted on July 2013).

[7] Omidvar, M., Li, X. Mei, Y. Yao, X., "Cooperative Co-evolution with Differential Grouping for Large Scale Optimization," *IEEE Transactions on Evolutionary Computation*, (accepted on 21 May 2013).

[8] Mohammad Nabi Omidvar, X. Li, and X. Yao, "Cooperative co-evolution with delta grouping for large scale non-separable function optimization,"

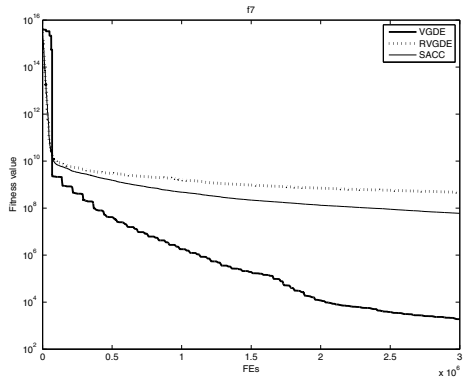

Fig. 2. The convergence curve of three algorithms on f7

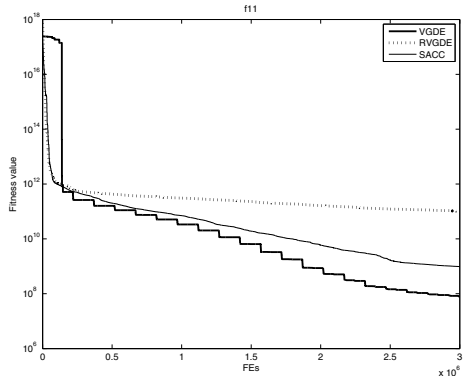

Fig. 3. The convergence curve of three algorithms on f11

in: Proceedings of IEEE Congress on Evolutionary Computation, pp. 1762-1769, 2010.

[9] R. Rach, J. S. Duan, "Near-field and far-field approximations by the Adomian and asymptotic decomposition methods," *Applied Mathematics and Computation*, vol. 217, no. 12, pp. 5910-5922, Feb. 2011.

[10] S. Ivvan Valdez, Arturo Hernndez, Salvador Botello, "A Boltzmann based estimation of distribution algorithm," *Information Sciences*, vol. 236, no. 1, pp. 126-137, Jul. 2013.

[11] Fei Wei and Yuping Wang, A new filled function method with one parameter for global optimization, *Mathematical Problems in Engineering*, vol. 2013, pp. 1-12, 2013.

[12] X. Li, K. Tang, M. Omidvar, Z. Yang and K. Qin, "Benchmark Functions for the CEC'2013 Special Session and Competition on Large Scale Global Optimization," *Technical Report, Evolutionary Computation and Machine Learning Group, RMIT University, Australia*, 2013.

[13] J. Vesterstrom and R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical," *in: Proceedings of the 2004 Congress on Evolutionary Computation*, pp. 1980-1987, 2004.

[14] R. Gamperle, S. Muller, and P. Koumoutsakos, "A parameter study for differential evolution," *in: Proceedings of WSEAS International Conference on Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*, pp. 293-298, 2002.

[15] Z. Yang, K. Tang, and X. Yao, "Self-adaptive differential evolution with neighborhood search," *in: Proceedings of IEEE World Congress on Computational Intelligence*, pp. 1110-1116, 2008.

[16] J. Nocedal, and S. J. Wright, *Numerical Optimization.* Springer-Verlag, New York, 1999.

[17] F. Wei, Y. P. Wang and Y.l. Huo, "Smoothing and Auxiliary Functions Based Cooperative Coevolution for Global Optimization," *in: Proceedings of IEEE Congress on Evolutionary Computation*, pp. 2736-2741, 2013.

[18] Antonio LaTorre, Santiago Muelas, Jos Marła Peña, "Large scale global optimization: Experimental results with MOS-based hybrid algorithms," *in: Proceedings of IEEE Congress on Evolutionary Computation*, pp. 2742-2749, 2013.