A Preferred Learning Based Adaptive Differential Evolution Algorithm for Large Scale Optimization

Xinran Ma, Jinliang Ding
State Key Laboratory of Synthetical Automation for Process Industries
Northeastern University
Shenyang, P.R. China
xrma_neu@163.com; jlding@mail.neu.edu.cn

***Abstract*—With the help of the cooperative co-evolution, differential evolution (DE) has been applied successfully from low-dimensional problems to large scale optimization. In this paper, we propose a preferred learning cooperative coevolution DE algorithm (LDECC-DG) which focuses on the basic optimizer for large scale optimization using cooperative coevolution. The proposed LDECC-DG builds on the differential evolution with cooperative coevolution and differential grouping (DECC-DG) algorithm which possesses an accurate grouping method and an effective basic optimizer method for large scale optimization. A novel DE algorithm called preferred learning based adaptive DE (LDE) is designed as a basic optimization algorithm for large scale problems and the control parameters in LDE are selected according to the self-adaptive strategy which corresponds to the preferred learning strategy. We show that how the LDE can improve the performance of Cooperative Co-evolution framework on account of its effectiveness. In order to evaluate the performance of LDECC-DG for large-scale global optimization, we carried out numerous computational studies on the CEC 2010 benchmark functions. The results show advantages of the LDECC-DG in both solution quality and convergence rate compared to other algorithms.**

Keywords—Large scale optimization; Cooperative co-evolution; Preferred learning adaptive strategy; Differential evolution

I. INTRODUCTION

Evolutionary optimization has shown a great advantage on many global numerical optimization problems for many years [1]. However, in science and engineering domain, most evolutionary algorithms (EAs) are often fail to find good near optimal solutions because of the huge number of decision variables, which called "curse of dimensionality". Therefore, many approaches have been introduced to EAs for solving large scale optimization. One of these approaches is called divide-and-conquer strategy [2]. The original large-scale problem will be decomposed into a set number of smaller and simpler subproblems, which is more operable and easier to optimize [3]. Then we can obtain the solution by cooperating with each subcomponent optimized solution. The techniques of decomposition have been verified in many classical optimization algorithms [4]-[7].

The cooperative co-evolutionary genetic algorithm (CCGA) is proposed by Potter and De Jong [8]. We can obtain an overall solution by cooperatively coevolving each subpopulation in lower dimensionality. The performance of the algorithm in dealing with large scale problems significantly depends on decomposition strategy and deteriorates rapidly when meets the nonseparable functions. Generally speaking, designing CCEAs which is able to tackle large-scale optimization (more than 100) in particular nonseparable problems is a necessary direction of the research.

Differential evolution (DE) is an efficient algorithm and DE is outstanding in comparison to PSO and EAs on the majority of the numerical benchmark problems [9]. Yang et al. [4] attempt to apply DE to cooperative coevolution (CC) model in the early stage and differential evolution with cooperative coevolution and random grouping (DECCG) was developed. Although DECCG was tested on functions of up to 1000 dimensions, the decomposition strategy of random grouping still has space to improve. Then the multilevel cooperative coevolution (MLCC) [10] was proposed by Yang et al., in which improved DECCG from the perspective of framework. An automatic way of decomposing called differential grouping was developed by Omidvar et al. [5] that designed CC model from the way of decomposition. The proposed algorithm DECC-DG has been evaluated using CEC'2010 benchmark functions and it has shown the superiority among other algorithms by increasing the accuracy of grouping interacting variables. However, the algorithms above did not improve CC model by designing subproblems optimization algorithm for large scale optimization.

This paper introduces a preferred learning based adaptive differential evolution algorithm (LDE) for cooperative coevolution, which enhances the performance of large scale optimization particularly for nonseparable functions via the design of novel subcomponent optimization. The proposed LDECC-DG builds on the success of DECC-DG, it adopts a preferred learning and adaptive strategy to help scaling up DE's performance for solving problems with a large number of decision variables. The main work of this paper is embodied in the following aspects.

- A powerful DE algorithm (LDE) which adopts preferred learning and self-adaptive strategies is proposed for large scale optimization using cooperative coevolution framework and analyzes why LDE is a good optimizer for subproblems.

- It is proofed in this paper that LDE can help coevolutionary optimization significantly. In fact, our new algorithm (LDECC-DG) is capable of tackling nonseparable problems with up to 1000 dimensions.

The rest of this paper is organized as follows. Section II gives a brief review of the related work on cooperative coevolution, differential grouping and classical differential evolution. Section III presents the proposed LDE and analyzes its effectiveness for solving large scale optimization. In Section IV, the experiments on our new algorithm LDECC-DG have been carried out. The performance of the LDE is compared to five state-of-the-art algorithms. Then the effectiveness of LDECC-DG is investigated by comparing with other two algorithms. Finally, Section V summarizes and concludes the paper.

II. Related Work

A. Cooperative Coevolution

Cooperative coevolution (CC) is a general framework based on decomposition strategy for large and complex problems. Potter and Jong [8] first applied CC into genetic algorithm for function optimization called CCGA, where the algorithm was tested on six test functions of up to 30 dimensions. But the CC framework was not used in higher dimensional problems. Recently, the framework of using CC in optimization has appeared which incorporated several algorithms such as evolutionary programming [11], evolution strategies [12], DE [4], [5], [7], and PSO [6].

The core idea of CC is to decompose a complicated high-dimensional problem into some low-dimensional subcomponents that easy to manage. It evolves these subcomponents in a cooperative way by cycles which include one complete process of evolution for all subcomponents within a fit value of fitness evaluations (FEs). We obtain the optimal solution by coadapting over all subcomponents. The size of each subproblem was determined by the capacity of DEs. However, there is some weakness exist in the CC algorithms for optimization. One is the nonseparable problems significantly decided by decomposition strategies and it did not perform well when variables were interdependent. Second, it is unknown that whether the optimizer is suitable for the entire CC framework or not.

B. Differential Grouping

Because of the qualities of CC algorithms were seriously relied on decomposition methods. Many researches have been studied on developing decomposition strategies. More recently, the grouping using an automatic decomposition strategy is incorporated on a CC framework. Linkage method such as random grouping was proposed by Yang et al. [4], it is designed to change grouping structures dynamically and increase the possibility of interacting variables into a set of subcomponents. Ideally, we should form the subcomponents according to some evidence which can identify interacting decision variables. Differential grouping [5] is an effective way to identify interdependence between two decision variables and group interactive variables together accurately via Theorem 1.

Theorem 1:

Define

$$\Delta_{\delta,x_p}[f](\vec{x}) = f(...,x_p+\delta,...) - f(...,x_p,...) \tag{1}$$

(1) refers to when given by an interval δ, the forward difference off with respect to variable x_p.

Let $f(\vec{x})$ be an additively separable function.

$\forall a, b_1 \neq b_2, \delta \in R, \delta \neq 0$, if the following condition holds:

$$\Delta_{\delta,x_p}[f](\vec{x})\Big|_{x_p=a,x_q=b_1} \neq \Delta_{\delta,x_p}[f](\vec{x})\Big|_{x_p=a,x_q=b_2} \tag{2}$$

then x_p and x_q are nonseparable.

During the grouping stage the *i*th and *j*th dimensions are examined by grouping function (2) which derived from Theorem 1, and then the subcomponents are formed accordingly.

$$\begin{aligned} &f(x_i+\delta_i,x_j) - f(x_i,x_j) \neq \\ &f(x_i+\delta_i,x_j+\delta_j) - f(x_i,x_j+\delta_j) \end{aligned} \tag{3}$$

In the optimization stage, the optimizer should exploit the provided grouping information and subcomponents are optimized in a round-robin fashion for a predetermined FEs. Consequently, more computational resources will be allocated for subcomponent with a higher contribution to the global fitness. However, the interdependencies between the subcomponents should keep to a minimum for the requirements of effective estimation. Actually, differential grouping can guarantee great accuracy which the interacting variables are placed within the same subcomponents for the most of the benchmark functions [5].

C. Differential Evolution

Differential evolution (DE) is a global optimization algorithm based on population evolution. Despite the simplicity of DE, it has the features of less greedy compared to other EAs algorithm which makes it fit to high-dimensional optimization.

In this paper, DE is used to solve large scale optimization as a subcomponent optimizer which needs more exploration due to the large search space [7]. In addition, it also needs to maintain its ability of exploitation. Therefore, the main emphasis of our research is getting a good balance between exploration (solution quality) and exploitation (convergence rate) by choosing strategies and setting parameters [13].

The mutation operation is the most significant step to obtain a mutant individual $v_{i,g}$ as shown in (4). The capability of mutation operation decides the searching direction and the searching area.

$$V_{i,g} = X_{i,g} + F*(X_{r1,g} - X_{r2,g}) \tag{4}$$

where F is called the mutation factor (scale factor). $X_{r1,g}$, $X_{r2,g}$ are considered as two random individuals which are different from mutation individuals. $X_{i,g}$ refers to a parent individual.

DE need to set a few control parameters: the population size NP, the mutation factor (scale factor) F, and the crossover probability crossover rate (CR). DE's control parameters F and CR are very sensitive to its setting techniques. Self-adaptive differential evolution with neighborhood search (SaNSDE) [14] is a novel DE with self-adapting F, CR and mutation strategy. In recent research of cooperative coevolution algorithm for large scale optimization, SaNSDE is widely used to be an optimizer [4], [5], [7] because of its better performance than other similar DEs.

III. IMPROVERD DIFFERENTIAL EVOLUTION ALGORITHM

A. Preferred Learning based Differential Evolution (LDE)

In the mutation operator of DE, we usually use a three-parameter notation DE/a/b to present the evolutionary operation [15]. "a" refers to the option of vectors and "b" denotes the number of mutation vectors in mutation operation. There are five widely used mutation operations DE/best/1, DE/rand/1, DE/current-to-best/1, DE/best/2, and DE/rand/2. These mutation strategies have their own base population vectors and members of difference population vectors. The DE/best/differ performs better in convergence rate. However, it is easy to premature convergence because the mutation operation effects on a local search space around the best individual. On the contrary, the DE/current/differ, because of the base vector is implemented as a target (current) individual who possesses a high diversity, it performs well in global search. As for the operator DE/current-to-best/1, it combines a current individual and a best individual as the basic individual and the difference individual respectively so that it does not achieve the best properties neither convergence or diversity. DE/rand/differ specializes in robustness with outstanding performance in the multimodal functions, especially at the later period of iteration [13].

All the above mutation operations only contain a single characteristic. But for large scale optimization, a good mutation operation means having both exploration (solution quality) and exploitation (convergence rate) [13]. For the purpose of obtaining a better performance may need a composite trial vector which integrates the information of diverse individuals and rapidity individuals. In addition, by incorporating a random element in the mutation operation is also essential for its effectiveness in dealing with multimodal functions [13]. As a result, many state-of-the-art DEs incorporate adaptive strategies or combination techniques to obtain the composite trial vector [16].

Cheng and Jin [17] first introduced social learning to PSO and the results showed a promising performance in solving large scale problems. By considering the analysis above, this paper develop a composite vector from the perspective of the mutation operation by social learning mechanism. An imitator vector will learn from the better individual vector and the new mutation operation is shown as follows.

$$V_{i,g} = X_{i,g} + F*(X_{better} - X_{i,g}) \tag{5}$$

where X_{better} refers to the individuals better than the current individual.

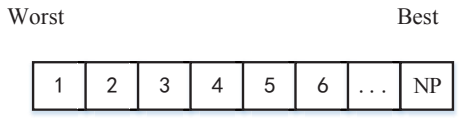


Fig.1 Population sorting

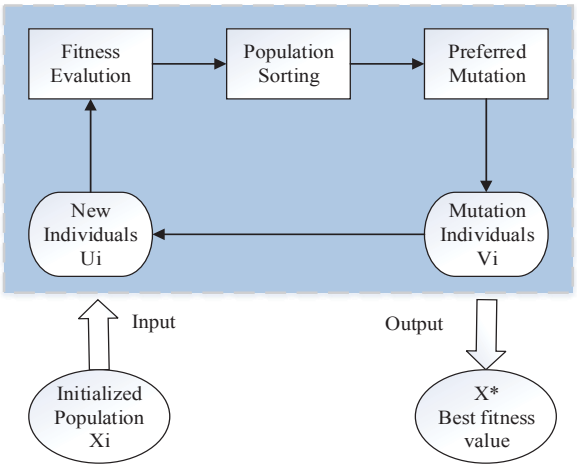


Fig.2 Main components of the proposed learning mechanism

Similar to the classic DE, the proposed algorithm initializes a population containing NP individuals. During the stage of fitness evaluation, every individual get a fitness value according to the objective function. The population is then sorted with a decreasing order of the individual fitness values, for a given individual (except for the last or call it the best individual), it learns from the other better individuals randomly where the model individual is expresses as X_{better}, as shown in Fig.1. The process of the learning mechanism is shown in Fig.2. For instance the individual 1, can learn from the second individual to the NPth, while the individual 2 can learn from the range from 3 to NP. Similarly, the individual m have the number of the individuals to learn is $m+1$-NP.

In the learning progress above, the mutation operation $V_{i,g}$ incorporates both rapidity information and diversity information. In addition, the information of random is introduced by the self-adaptive strategy as the next section.

B. The Adaptive Strategies for LDE

Considering the analysis of advantages of using abundant information for evolving, we adopt self-adaptive strategy for to form composite trial vectors. As point out above, we introduce preferred learning strategy to obtain both convergence and diversity. In the adaptive strategies for LDE, we employ DE/rand/1 as shown in Eq.(6) to supply the random information.

$$V_{i,g} = X_{r1,g} + F*(X_{r2,g} - X_{r3,g}) \tag{6}$$

It is proofed in [13] that DE/rand/1 is a robustness strategy

which is effective to the multimodal functions. Therefore LDE selects Eq.(6) along with preferred learning mutation operation Eq.(5) to enrich the information of population, and the new generated trial vector is shown below:

$$V_{i,g} = \begin{cases} X_{i,g} + F*(X_{better} - X_{i,g}), \text{ if } U_i(0,1) < p \\ X_{r1} + F*(X_{r2,g} - X_{r3,g}), \quad otherwise \end{cases} \quad (7)$$

Algorithm 1: LDECCDG-DG (s, cycles, FEs)

```
groups ← differential grouping
pop (1:NP, 1:n)← random pop
(best,best_val) ←evaluate (pop)
for i=1 to cycles do
 for j=1 to size (groups) do
    indices ← groups [j]
    subpop ← pop [:,indicies]
    subpop ← LDE (best, subpop, FEs)
     while  iter<itermax  //LDE algorithm
      do
       Arrange the current population individuals in
       descending order of their fitness values.
      for i=1 to NP
         Randomly choose four different individuals x_{o,g},
         x_{r1,g}, x_{r2,g}, and x_{r3,g.}
         Set o=i when g ≤ g_max
         To form the F_o for the individual x_{o,g} as:
         F_o = N_i((NP−o)/NP, 0.1)
         To form mutant vector V_{i,g} via the self-adaptive
         strategy  as:
          V_{i,g} = { X_{i,g} + F_o*(X_better − X_{i,g}), if U_i(0,1) < p
                    { X_{r1} + F_o*(X_{r2,g} − X_{r3,g}),  otherwise
      end for
      for i=1 to NP
         Generate  CR_i = N_i((NP−i)/NP, 0.1)
         for the target individual x_{i,g}. Generate trial vector
         u_{i,g} via binomial Crossover as follow:
         for j=1 to D
          u_{i,g}(j) = { v_{i,g}(j), if (rand_j[0,1] ≤ CR) or (j = j_rand), j = 1,2,...,n
                       { x_{i,g}(j), otherwise
         end for
      end for
       Evaluate the trial vector u_{i,g}
         for i=1 to NP
            x_{i,g+1} = { u_{i,g}, if (f(u_{i,g}) ≤ f(x_{i,g}))
                        { x_{i,g},  otherwise
         end for
        iter=iter+1
   end while
pop [:, indicies] ←subpop
(best, best_val) ← evaluate (pop)
 end for
end for
```

where p controls which mutation strategy to use. It adopts a learning strategy to employ different mutation operation for different evolution stages. The specific set of parameter p can be found in [14].

Different parameter settings may influence its characteristics and different strategies adopted for each operation can obtain different searching features which is fit for different scales and problems [13], [18]. In addition, different distributions of mutation and control parameters determine the quality of exploration and exploitation. Tang et al [13] first attempted to set control parameters with an adaptive strategy called IDP setting method that accord with the differences between individual's fitness values. They noticed that the individuals with better fitness values are more close to the global optimal, whereas individuals with worse values are far away from it. Based on the idea of IDP, our preferred learning strategy also adapt the setting of F and CR by the order of fitness values in Section III-A so that the solutions with good qualities are assigned with smaller values to execute local search. As a consequence, solutions with good qualities are assigned with small values to execute local search and those poor ones are assigned with larger values to jump out of the poor space for better exploration. Some more details of LDE using this principle are summarized as follows.

Scale factor F setting: The individual x_i is supposed to the ith most superior one.

$$F_i = \frac{NP - i}{NP} \quad (9)$$

$$CR_i = \frac{NP - i}{NP} \quad (10)$$

where the control parameter F_i and CR_i associated with the target individual x_i. Then we randomize the parameters with a Gaussian random number, which the mean is the original value and the standard deviation 0.1. The settings can be modified as:

$$F_i' = N_i(F_i, 0.1) \quad (11)$$

$$CR_i' = N_i(CR_i, 0.1) \quad (12)$$

The role of randomization is to maintain search efficiency when the control parameters are near their mean values, and the distant ones can develop the diversity of search. If the parameter value we extracted from the Gaussian distribution is beyond the range of (0,1), it will not exert an influence and a new value originated from Gaussian distribution will instead of it until the parameter value is within normal range.

C. LDE under the CC Framework

Having LDE as the basic optimizer for subproblems, it is explicit to design the LDE with cooperative coevolution, called LDECC-DG, by the following step of DECC-DG [5]. The pseudocode of LDECC-DG is shown in Algorithm 1.

IV. EXPERIMENTIAL STUDIES

In this section, we first verify the performance of LDE compared to several state-of-art DE algorithms, and demonstrate why LDE is fitter for large scale optimization using cooperative coevolution. Then we compare LDECC-DG with DECC-DG and DECC-G on large scale global benchmark functions.

A. Effects Evaluation of LDE

It is obviously that a large scale problem will be decomposed into low-dimensional subproblems under the CC framework. Therefore, an effective optimizer need to have better performance on medium dimensional (not more than a hundred) problems, meanwhile the computing time of each optimizer should not be too long. In this section, we demonstrate the LDE is provided with both advantages above by some empirical evidence. Table I shows the five state-of-the-art DE algorithms [14], [20]-[23] and a particle swarm optimization (PSO) [24] we used for empirical studies. We compared these algorithms with LDE on 25 test functions proposed in the CEC'05 special session on real-parameter optimization [25] by a statistical conclusion using 30 independent runs. A two-tailed t-test was used to investigate if there is a difference between LDE and other six algorithms. If the p-value is smaller than level 0.05, we will reject the null hypothesis. The p-value is set to 0.05 in our experiment. "+/-" means LDE has "better/poorer" performance with significant difference and "=" represents there is no difference between the two algorithms from the aspect of the significance level. We can obtain the statistical results on Table II.

TABLE I. DESCRIPTION OF COMPARISON ALGORITHMS

Algorithms	Description
SaDE	Self-adaptive Differential Evolution
CoDE	Differential Evolution with Composite Trial Vector Generation Strategies and Control Parameters
jDE	Self-adapting Control parameters in Differential Evolution
JADE	Adaptive Differential Evolution with Optional External Archive
SaNSDE	Self-adaptive Differential Evolution with Neighborhood Search
CLPSO	Comprehensive Learning Particle Swarm optimization

TABLE II. STATISTICAL COMPARISON RESULTS OF LDE

Algorithms	Dimensions of the Benchmark Functions									Average Time for Each Function
	10D			*30D*			*50D*			
	+	=	-	+	=	-	+	=	-	**50D/s**
SaDE	6	16	3	8	14	3	7	16	2	528.01
CoDE	6	14	5	9	11	5	6	18	1	332.82
jDE	12	10	3	11	12	3	7	16	2	750
JADE	5	14	6	4	15	6	6	14	5	650.04
SaNSDE	5	17	3	8	14	3	9	13	4	363.97
CLPSO	11	12	2	12	10	3	13	10	2	223.44

From the Table II, we can see the advantage of LDE among the other algorithms and its average computation time for each function on 50D only 354.62s. Then we choose several algorithms with better performance (CoDE, JADE and SaNSDE) as candidates to further compare the performance with LDE. Four typical functions f2, f6 f10, and f13 are chosen to further illustrate the performance of LDE. As shown in Figs.3, for the choosing functions up to 50 dimension, JADE seems to obtain better results. However, LDE performs steadily on the same function with different dimensions or in other words it can maintain better performance on the higher dimension. In addition, the time-consuming of JADE is nearly twice to LDE. Consider about comprehensive factors above (performance and time-consuming), LDE is more suitable for cooperative coevolution framework as an optimizer.

B. Parameter Settings for LDECC-DG Evaluation

For purpose of testing the performance of LDECC-DG, we choose the CEC'2010 special session on large-scale global optimization [19]. A set of 20 benchmark test functions is classified into five groups as Table II.

As for the population size we used 50 as DECC-DG [5]. All of the experimental results are recorded base on 25 independent runs. The maximum number of fitness was set to 3e+06 as suggested in [5]. When we executed grouping operation, the value of ϵ was set to 10^{-3}. For this research, n and m were set to 1000 and 50, respectively. The t-test experiment setting is the same as Section IV-A.

TABLE III. DESCRIPTION OF CEC'2010 BENCHMARK FUNCTIONS

f1-f3	Separable functions
f4-f8	Single-group m-nonseparable functions
f9-f13	$\frac{n}{2m}$ -group m-nonseparable functions
f14-f18	$\frac{n}{m}$ -group m-nonseparable functions
f19-f20	nonseparable functions

[a.] n refers to dimensionality of the problem and m refers to the number of variables in each nonseparable subcomponent.

C. Results on CEC'2010 Benchmark Functions

In this section, we evaluate the performance of LDECC-DG through comparing it with two state-of-the-art Cooperative Co-evolution DE algorithms for large scale optimization. One of the comparing algorithms DECC-DG uses SaNSDE as a subcomponent optimizer with Differential Grouping which is the same as LDECC-DG and the other DECC-G is similar to DECC-DG except that it adopts the random grouping strategy.

As shown in Table IV, we can discover that LDECC-DG performed significantly better than the other two algorithms. The performance of DECC-DG and LDECC-DG outperformed DECC-G on most nonseparable functions as the grouping accuracy enhanced. On closer inspection, we can see that LDECC-DG achieved better results than DECC-DG on nonseparable functions f5, f7, f8, f9, f10, f11, f12, f13, f15, f17, f18 and f20. What is noteworthy is that LDECC-DG is even worse than DECC-G on f13, f18 and f20 although it has

already modified the performance of them dramatically compared to DECC-DG.

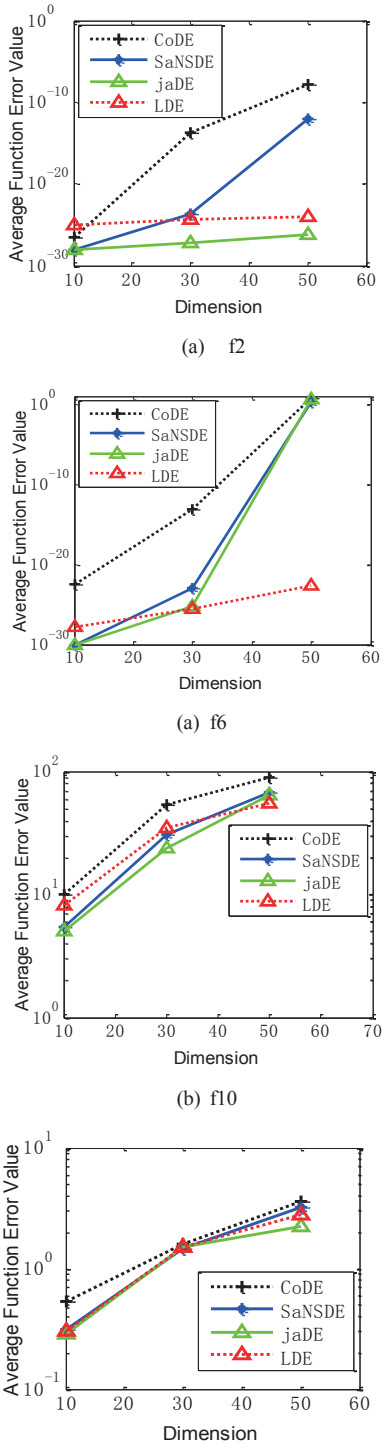
We choose the 12 functions with better performance to further understand the entire search process. Fig. 4 shows the convergence processes of these functions and it demonstrates that LDECC-DG not only achieved the best results, but also has faster convergence. Therefore, we can confirm that LDE is significantly effective as an optimizer for cooperative co-evolution. In particular, for those functions which superior individuals are closer to the global solution such as schwefel's problem 1.2 (f7, f12 and f17) and rosenbrock's function (f13, f18 and f20), the preferred learning mutation operation and adaptive strategies of LDE may help the algorithm to explore the optimal area rapidly and accurately without the restriction of grouping strategy.

TABLE IV. COMPARISION BETWEEN LDECC-DG, DECC-DG AND DECC-G ON THE CEC'2010 BECHMARK FUNCTIONS.

Functions	DECC-G	DECC-DG	LDECC-DG	p-value
f1 mean	**1.41E-09**	2.99E+06	4.20E+01	2.37E-31
std	9.24E-10	1.87E+06	1.10E+00	
f2 mean	**4.85E+02**	4.22E+03	5.77E+03	6.23E-11
std	1.84E+01	1.93E+02	2.60E+02	
f3 mean	2.40E+01	**1.08E+01**	1.78E+01	3.84E-26
std	1.80E+00	4.28E-01	1.37E-02	
f4 mean	6.08E+12	**5.59E+11**	4.35E+12	1.72E-14
std	5.92E+11	1.63E+11	7.46E+12	
f5 mean	2.95E+08	6.91E+07	**6.80E+07**$^{+}$	1.56E-03
std	4.56E+07	8.45E+06	6.01E+06	
f6 mean	4.18E+06	**1.64E+01**	**1.80E+01**$^{=}$	1.03E-01
std	1.44E+06	1.10E+00	2.03E+01	
f7 mean	9.21E+05	6.01E+04	**2.26E+03**$^{+}$	1.17E-03
std	1.89E+04	2.77E+04	4.05E+03	
f8 mean	8.39E+07	6.53E+07	**3.81E+07**$^{=}$	2.36E-01
std	3.36E+07	4.77E+06	2.97E+05	
f9 mean	1.60E+08	6.67E+08	**2.61E+08**$^{+}$	1.00E-11
std	1.32E+07	1.86E+08	2.83E+07	
f10 mean	8.83E+03	7.64E+03	**6.90E+03**$^{+}$	3.94E-15
std	4.72E+02	2.74E+02	1.53E+02	
f11 mean	2.71E+01	3.26E+01	**2.64E+01**$^{+}$	7.04E-18
std	4.43E+00	1.17E+00	2.12E+00	
f12 mean	3.42E+04	2.30E+04	**1.72E+02**$^{++}$	4.16E-19
std	7.51E+03	6.97E+03	2.44E+02	
f13 mean	**5.74E+03**	3.03E+07	3.25E+05	2.44E-17
std	2.52E+03	8.54E+06	1.60E+05	
f14 mean	4.74E+08	**2.22E+07**	1.85E+08	3.74E-31
std	2.50E+07	2.14E+06	5.63E+06	
f15 mean	6.42E+03	3.44E+03	**2.51E+03**$^{+}$	3.86E-08
std	2.91E+03	1.20E+02	1.84E+02	
f16 mean	1.01E+02	2.00E+01	**1.03E+01**$^{+}$	1.99E-13
std	1.10E+01	1.39E+00	1.02E+00	
f17 mean	3.27E+05	1.32E+04	**4.32E+03**$^{+}$	1.30E-26
std	4.59E+03	2.28E+00	2.18E+02	
f18 mean	**3.32E+07**	7.31E+10	3.20E+08	5.26E-11
std	8.88E+06	2.38E+09	1.36E+07	
f19 mean	1.86E+06	**1.35E+06**	1.83E+06	1.54E-24
std	2.63E+05	9.54E+04	1.24E+05	
f20 mean	**3.54E+05**	1.24E+09	1.27E+07	3.31E-18
std	5.20E+04	5.10E+08	1.40E+07	
+/-/=	11/7/2			

b. "++" represents LDECC-DG performs much better than DECC-DG.

For some functions such as the rotated function, the superior individuals are not always around the global solution [13]. We also can obtain good performance because of the adaptive strategies in LDE. With the randomization of F and CR, the search process has potential on both effectiveness and efficiency. As a consequence, some rotated functions such as f5, f6, f10, f11, f15 and f16 tested in LDECC-DG are not inferior to DECC-DG and even achieve better performance.



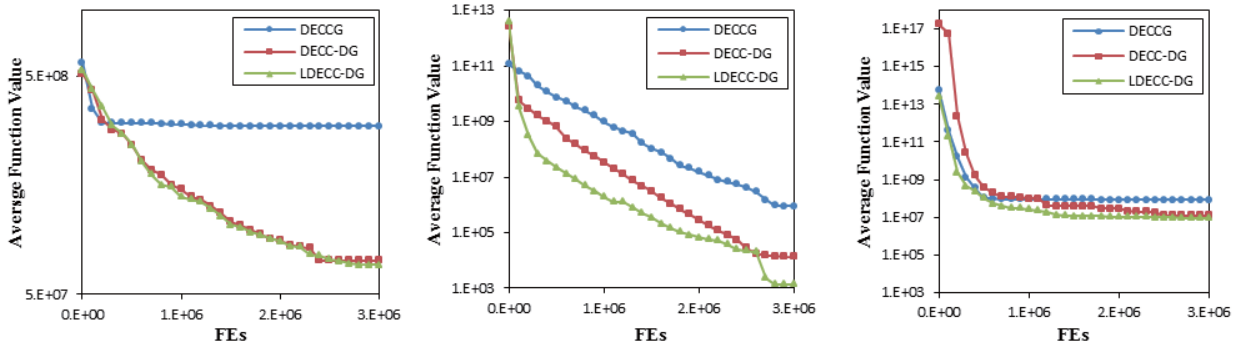
(a) f2

(a) f6

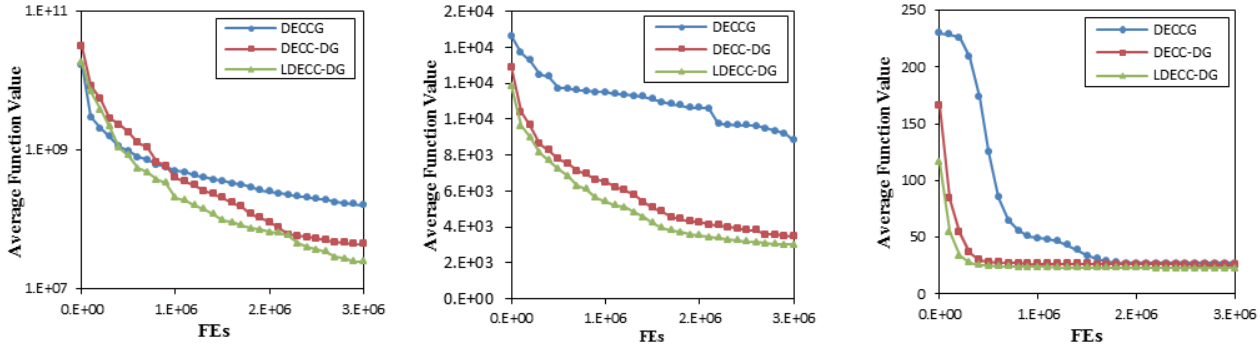
(b) f10

(c) f13

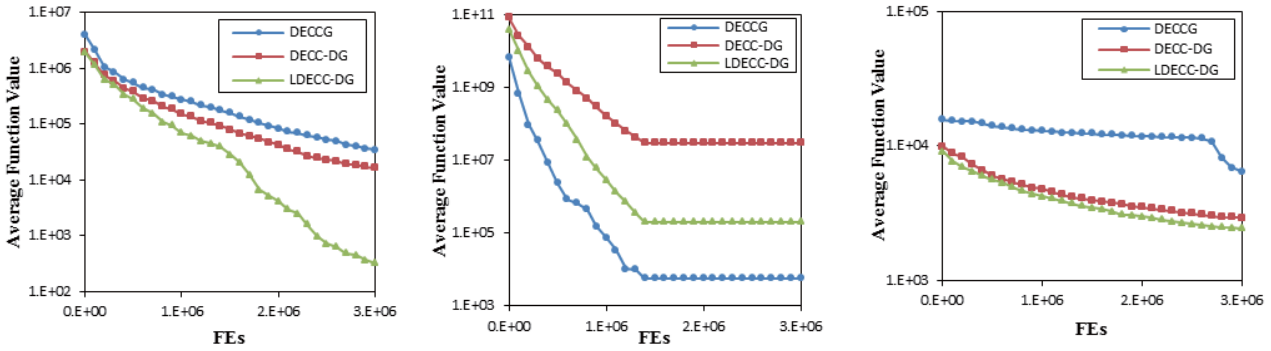
Fig.3. 25 averaged function error values of four DEs on f2, f6, f10 and f13 of 10, 30 and 50 dimensions.



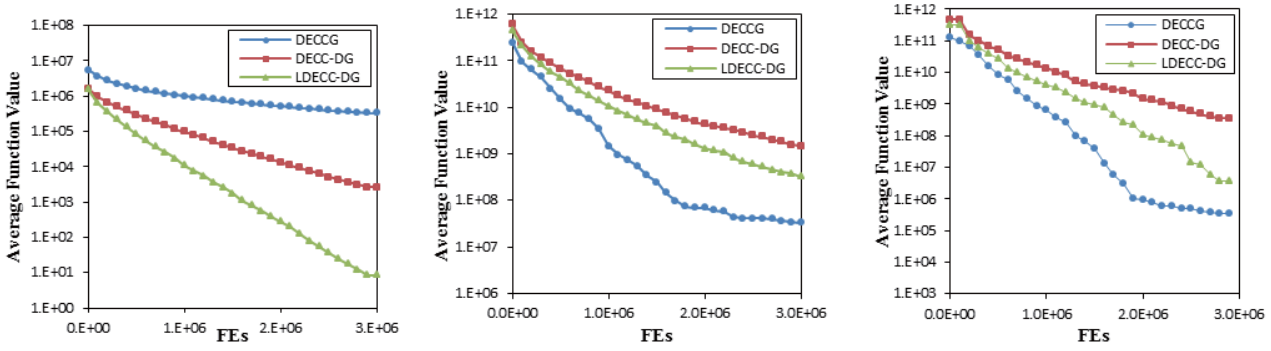
(a) f5 (b) f7 (c) f8



(d) f9 (e) f10 (f) f11



(g) f12 (h) f13 (i) f15



(j) f17 (k) f18 (l) f20

Fig.4. The evolution convergence plot with the mean best values on f5, f7, f8, f9, f10, f11, f12, f13, f15, f17, f18 and f20, where the results were originated from the average of 25 independent runs.

V. Conclusions

This paper proposes a novel differential algorithm for large scale optimization problems. For subproblem optimization, it is a promising way to use LDE which adopts preferred learning adaptive strategies. Some theoretical and experimental analyses were given to demonstrate how such strategies can help LDE to enhance the ability of exploration. With the help of LDE, we proposed a new CC optimization algorithm, LDECC-DG, for large scale problems. Extensive quantities experimental studies were carried out to evaluate the performance of LDECC-DG on CEC'2010 benchmark functions.

In order to further understand the performance of LDECC-DG, we first demonstrated why LDE is more competitive to other variants of DE as a basic optimizer. Then SaNSDE was compared with LDE by using cooperative coevolution for large scale optimization. The comparison was realized by comparing our whole algorithm LDECC-DG with DECC-DG and DECC-G. The experimental results showed that LDE can greatly improve the performance of LDECC-DG and it was more effective than other two algorithms. It also revealed that LDECC-DG was more competitive on high-dimensional nonseparable functions especially it covered the shortage of schwefel's problem 1.2 and rosenbrock's function through the valid exploration ability provided by preferred-learning mutation operation and adaptive strategies compared to DECC-DG. The convergence plots which describe the evolution process of the benchmark functions also support the correctness of our point. Furthermore, all the results confirmed our analysis that LDE is a more competitive optimizer for cooperative coevolution and LDECC-DG is very effective and efficient in tackling large scale optimization problems with dimensions up to 1000.

In the future, we will focus on the intelligence grouping methods with the tendency of automation in order to coordinate the subcomponent optimization for obtaining the better performance. We also would like to try applying LDECC-DG to process industry optimization such as optimization of crude oil distillation units, in order to ascertain its true potential as a valuable optimization technique for real-world optimization.

Acknowledgment

This work was supported in part by National Natural Science Foundation of China Projects under Grant 61525302 and Grant 61590922, and in part by the Projects of Liaoning Province under Grant 2014020021 and Grant LR2015021. The author would like to thank M.N. Omidvar for providing the code of DECC-DG.

References

[1] R. Sarker, M. Mohammadian, X. Yao, Evolutionary Optimization, Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[2] R. Descartes, Discourse on Method, 1st ed. Prentice Hall, Upper Saddle River, NJ, USA, Jan. 1956.

[3] G.B. Dantzig and P. Wolfe, "Decomposition principle for linear programs," Oper. Res. vol.8, no. 1. pp. 101-111, 1960.

[4] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," Inform. Sci., vol. 178, no. 15, pp. 2986-2999, August 2008.

[5] M. N. Omidvar, X. Li, Y. Mei, and X. Yao, "Cooperative co-evolution with differential grouping for large scale optimization," IEEE Trans. Evol. Comput., vol.18, no.3, pp. 378-393, June 2014.

[6] X. Li and X. Yao, "Cooperatively coevolving particle swarms for large scale optimization," IEEE Trans. Evol. Comput., vol. 16, no. 2, pp. 210-224, April 2012.

[7] M. N. Omidvar, X. Li, Z. Yang, and X. Yao, "Cooperative coevolution for large scale optimization through more frequent random grouping," in Proc. IEEE Congr. Evol. Comput., July 2010, pp. 1754-1761.

[8] M. A. Potter and K. A. De Jong, "A cooperative coevolutionary approach to function optimization," in Proc. Int. Conf. PPSN, 1994, vol. 2, pp. 249-257.

[9] V. Jakob, and R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," Evol. Comput., *2004. CEC2004*. Congr. on. vol. 2. IEEE, 2004.

[10] Z. Yang, K. Tang, and X. Yao, "Multilevel cooperative coevolution for large scale optimization," in Proc. IEEE Congr. Evol. Comput. June 2008, pp.1663-1670.

[11] D. Bertsekas, Nonlinear Programming (Optimization and Neural Computation Series). Belmont, MA, USA: Athena Scientific, 1995.

[12] R. Descartes, Discourse on Method, 1st ed. Prentice Hall, Upper Saddle River, NJ, USA, January. 1956.

[13] L. Tang, Y. Dong and J. Liu, "Differential evolution with an individual-dependent mechanism," IEEE Trans. on Evol. Comput., vol. 19, no. 4, pp. 560-574, August. 2015.

[14] Z. Yang, K. Tang, and X. Yao, "Self-adaptive differential evolution with neighborhood search," in Proc. IEEE Congr. Evol. Comput., June. 2008, pp. 1110–1116.

[15] R. Storn and K. V. Price, "Minimizing the real functions of the ICEC'96 contest by differential evolution," in *Proc. IEEE Int. Conf. Evol. Comput.*, Nagoya, Japan, May 1996, pp. 842–844.

[16] A. K. Qin, P.N. Suganthan, Self-adaptive differential evolution algorithm for numerical optimization, in: Proceedings of the 2005 IEEE Congr. Evol. Comput., vol. 2, pp. 1785–1791, 2005.

[17] R. Cheng and Y. Jin, "A social learning particle swarm optimization algorithm for scalable optimization," Inform. Sci. pp.43-60, 2015

[18] Y. Liu, X. Yao, Q. Zhao, and T. Higuchi, "Scaling up fast evolutionary programming with cooperative coevolution," in Proc. IEEE Congr. Evol. Comput., 2001, pp. 1101–1108.

[19] K. Tang, X. Li, P. N. Suganthan, Z. Yang, and T. Weise. (2009). Benchmark functions for the CEC'2010 special session and competition on large-scale global optimization. Nature Inspired Comput. Applicat. Lab., University of Science and Technology of China. Hefei, China. [Online]. Available:http://goanna.cs.rmit.edu.au/~xiaodong/publications/lsgocec10.pdf

[20] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," IEEE Trans. Evol. Comput., vol. 15, no. 1, pp. 55–66, February. 2011.

[21] J. Zhang and A.C. Sanderson, "JADE: adaptive differential evolution with optional external archive," IEEE Trans. Evolut. Comput., vol.13, no.5, pp. 945-958, 2009

[22] A. K. Qin, P.N. Suganthan, Self-adaptive differential evolution algorithm for numerical optimization, in: Proceedings of the 2005 IEEE Congr. Evol. Comput., vol. 2, 2005, pp. 1785–1791.

[23] Qin, A. Kai, V. Huang, and Ponnuthurai N. Suganthan. "Differential evolution algorithm with strategy adaptation for global numerical optimization," IEEE Trans Evol. Comput. vol.13 no.2 pp. 398-417, 2009

[24] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," IEEE Trans. Evol. Comput., vol. 10, no. 3, pp. 281–295, 2006.

[25] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," Nanyang Technol. Univ., Singapore, Tech. Rep. KanGAL #2005005, May 2005, IIT Kanpur, India.