Cooperative Coevolution Based Differential Evolution Embedded With Perfunctory Mechanism Guided Search Moves For Large Scale Global Optimization

Sayan Basu Roy, Mainak Dan, and Swagatam Das, *Senior Member, IEEE*

***Abstract*—Performance of majority of evolutionary and swarm real-parameter optimizers like Differential Evolution (DE), Particle Swarm Optimisation (PSO), real-coded Genetic Algorithms (GAs) deteriorate significantly with the rapid growth of the dimension of the problem at hand. Thus some special operators and/or search mechanisms are required along with those conventional algorithms to handle high dimensional optimization problems. This article attempts to modify Differential Evolutionwith Cooperative Coevolution technique for solving large-scale global optimization. We also incorporated a strategy, imitating the human psychology, resulting in an improved performance on large-scale benchmark functions. The proposed algorithm is applied on high dimensional benchmark problems of CEC (Congress on Evolutionary Computation) 2008 and 2010 and the obtained results are compared with those of 6 state-of-the-art algorithms (including one recently developed powerful algorithm, CCPSO2) under identical experimental conditions. The comparison tables clearly reflect statistically superior performance of our algorithm over those existing state-of-the-art algorithms on a wide variety of numerical benchmarks.**

Keywords*—Controlled Mutation, Cooperative Coevolution, Cauchy Distribution, Differential Evolution, Power mean, Random Grouping

I. Introduction

FOR several decades, attempts were made to solve mathematically intractable search and optimization problems by using meta-heuristics. So far developed swarm and evolutionary computing algorithms has achieved great success on many numerical and combinatorial benchmark problems in recent years[1]. However most of these algorithms lose their efficiencies as the dimensionality of the problem increases and goes beyond 100 or so. They face troubles in finding the global optimum with sufficient accuracy and without consuming too much Function Evolutions (FEs) in the large search space.

This phenomenon is not astonishing and is primarily caused by the increase of the search volume exponentially with dimension. Let us consider placing 100 points onto a real interval, say $[0, 1]$. To achieve a similar coverage, in terms of distance between adjacent points, in the $10D$ space $[0, 1]^{10}$ would require $100^{10} = 10^{20}$points. Naturally, the previously mentioned 100 points now appear as isolated points in the vast empty space. Thus the distance measures break down in higher dimensions and a search mechanism, that is effective in small scale, might be poor in large scale problems.

Many real world problems involve optimization of a huge number of variables, starting from just 100 variables may be up to 1000 variables and sometimes even more than that. A few typical examples of such problems are shape optimization[2],[3], high-dimensional waveform inversion[4], large scale Economic load dispatch [5] etc. In shape optimization a large number of shape design variables is used to represent highly complex structures such as turbine blades[3], aircraft wings[6] and heat exchangers[2].

Moreover, the expensive nature of these problems gets magnified. In many real parameter high dimensional problems the evaluation of the objective function involves interaction with different other modules like simulation software. RoboCup simulation software is a perfect example[2]. In the areas of shape optimization, Evolutionary Robotics, Multidisciplinary Design Optimization(MDO) the evaluation of the objective function is very costly as well as time consuming. That is why new intelligent techniques are required to cut down the total number of function evaluations without degrading the performance of the algorithms.

Many strategies have already been developed for tackling these kinds of large scale problems. The most popular of them are Cooperative Coevolution, micro EAs, local search based methods etc. Cooperative Coevolution, proposed by Potter *et al.*[7], follows a divide-and-conquer approach to divide the decision variables into smaller subpopulations and then each subpopulation is optimized separately and finally they are cooperated to give the whole vector for the function evaluation purpose. Micro approaches are also effective in this domain. They use a very small number of populations, equipped with very efficient exploration and exploitation schemes.

We have chosen Cooperative Coevolution strategy which is being embedded on the Differential Evolution[8] framework giving promising results. The sub-component optimizer DE variant, chosen in this paper, has recently appeared as a conference article[**?**] in a short version. Difficulties of Cooperative Coevolution (such as which variables to group together and also what should be the group size) are removed using some novel schemes (discussed in detail in section V). Also a strategy, imitating human psychology, is introduced. Experimental results reveal the efficiency of this scheme.

To judge the quality of the proposed algorithm, an extensive comparative study is taken into account on the benchmark functions from both CEC 2008[11] and 2010[12] competi-

tions. The performance of CCDE_PM is compared with six well known existing algorithms (such as CCPSO2[13], sep-CMA-ES[14], MLCC[15], DMS-PSO[16] etc.), revealing the strength of it as CCDE_PM outperforms the others in most cases.

However, the rest of the paper is organized in the following way. Section II provides a brief description of the existing methods to handle large scale problems. Section III outlines the Cooperative Coevolution strategy in detail. The DE variant used to optimize the subcomponents is described in an elaborate fashion in section IV. Section V is devoted for the detailed discussion of the novel schemes to tackle the drawbacks of Cooperative Coevolution. The strategy, imitating human psychology, is incorporated in section VI. Experimental settings along with a brief description of the benchmark suites are presented in section VII. Results, comparative study and analysis are made at section VIII. Finally section IX concludes the paper and unfolds the directions of future research.

II. Previous Works

This section is included to produce a brief introduction of the most prominent methods so far developed in literature for solving large scale optimization problems. The strategies can be roughly categorized into three classes: the Cooperative Coevolution methods, the micro EAs and the Local Search (LS) based methods.

A. Cooperative Coevolution Methods

This is the most well-known method to handle high dimensional optimization, utilizing an automatic divide and conquer strategy. A high dimensional objective vector is decomposed into smaller subcomponents that can be separately optimized by conventional EAs. But the performance of the CCEAs highly depends on the decomposition scheme. They show poor results especially in non-separable problems. To circumvent such difficulties, a new decomposition strategy based on random grouping is proposed by Yang *et al.*[17],[18]. The authors also introduced an adaptive weighting scheme to fine-tune the solutions. Li and Yao used this Cooperative Coevolution with PSO by integrating random grouping and adaptive weighting[19], showing improved performance. Omidvar *et al.*[20] exposed experimentally the fact that frequent random grouping gives beneficial improvement to the performance of CCEAs than the incorporation of adaptive weighting. Motivated from them, Li and Yao extended their CCPSO[19] in the name of CCPSO2[13] by eliminating adaptive weighting and incorporating a new PSO model using Cauchy and Gaussian distributions. Zamuda *et al*[21] modified DE by a log-normal self-adaption to control parameters along with Cooperative Coevolution as a decomposition mechanism.

B. Micro EAs

Micro EAs are another category of high dimensional optimization problems. These are instances of typical EAs, characterized by small population size and often simple fitness functions. Goldberg[22] applied this micro concept on GA using a population of only 3 individuals, showing nominal convergence. He embedded a process of elitism in his micro-GA. Krishnakumar[23] proposed another scheme of micro-GA with population of 5 individuals and applied it in large scale seismic trace matching problem[24]. Later Huang and Mohan used this micro concept on PSO, showing promising results on four benchmark functions of 1000 dimensions. The main drawback of these schemes is premature convergence due to inefficient exploration, which is primarily due to the small number of population, unable to cover the whole search space efficiently. To overcome this trouble, micro-EAs are generally combined with diversity preserving scheme to retain population diversity, thereby enhancing its exploration capacity. Sometimes, multiple restart mechanisms are also incorporated with micro concepts. For example, micro-PSO proposed in[25] is equipped with a repulsion technique based on the works of Parsopoulos and Vrahatis[**?**]. Following the trend, attempts are made to use the efficiency of Cooperative Coevolution with the micro concept. This idea is reflected on the work of Parsopoulos, who developed a Cooperative micro-PSO and Cooperative micro-DE for large scale global optimization.

C. Local Search based Method

Integrating a LS method with a global evolutionary optimizers, termed as Memetic Algorithms (MAs)[26], have been applied many times in high dimensional problems. Noman and Iba[27] embedded a crossover based LS, known as Fittest Individual Refinement (FIR) in the original DE framework to improve its performance in high dimensional domains. Zhao *et al.* applied a Quasi-Newton LS method with the DMS-PSO[16], to qualify the search mechanism of the latter. Tseng and Chen[28] developed a Multiple Trajectory Search (MTS) algorithm, where each search agent follows one of three proposed candidate LS methods. The agents choose the LS procedure that best fits the landscape of its neighborhood. Inspired from this, Zhao *et al.*[39](ABC) invented one hybrid algorithm of MTS and self-adaptive DE (SaDE) to refine solution frequently at different search stages meeting both global and local search requirements. Recently, Molina *et al.*[29] developed an algorithm, termed as MA-SW-Chains[29], ranked first in the CEC 2010[12] competition on large scale global optimization. This algorithm is nothing but a Memetic Algorithm based on the scalable Solis Wets LS method. After that, it is modified as MA-SSW-Chains[30], where the classical Solis Wets is replaced by a Subgrouping Solis Wets (SSW) algorithm.

D. Others

Apart from the above mentioned categories, some other prominent efforts are taken for tackling high dimensional problems. This approaches include population size variation[31],[32], opposition-based learning [33],[34], neighborhood search[35], hybridization[36] etc.

However, Cooperative Coevolution is chosen in our paper due to its success over wide variety of problems when applied with different evolutionary algorithms. The next section deals with a detailed discussion of Cooperative Coevolution scheme.

III. Detail View Of Cooperative Coevolution Strategy

As already stated, Cooperative Coevolution (CC) is a general framework for applying EAs to large and complex problem domains utilising the divide-and-conquer strategy. Potter and De Jong[**?**] applied this concept first time on Genetic Algorithm (GA). However, the original framework of CC is given below:

1) Decompose an n dimensional objective vecto into m s-dimensional subcomponents $(n = m * s)$
2) For $i = 1$ to m
3) Optimize the i-th subcomponent with a certain EA for a predefined number of function evaluations (FEs).
4) Stop if the halting criteria is satisfied otherwise go to step 2 and repeat the cycle.

The cooperation i.e. concatenation of subcomponents occurs only at the time of function evaluation. However, size of each subcomponent (s) should be within the optimization ability of the EA used.

Potter applied CC framework to concept learning and neural network construction[37]. Later, many other attempts are made with CC on neural network domain. Cao *et al.* shows another application of CC in the domain of pedestrian detection system[40].

But the most popular application of CC is high dimensional optimization. Many attempts have been made by the researchers to apply CC concept with variety of EAs over a few years. At the earlier stage, two simple decomposition schemes are used[38],[7],[39]. One is one-dimensional based and the other is splitting in half strategies. The first one breaks the high-dimensional vector into single variables. This strategy works well with separable problems but lacks quality performance on non-separable functions, as it does not consider inter-dependencies among variables. The splitting-in-half scheme divides the high dimensional vector into two equal halves, thereby reducing the problem dimension to half the original problem. But the problem is that, if n is large enough, $n/2$ would be still considerably large. In those cases, the performance is not up to the mark.

However, the basic problems, faced by this Cooperative Coevolution scheme, are pointed out below:

- What should be the subcomponent size?? In general, in case of the separable functions small subcomponent size results well, whereas the non-separable functions prefer comparatively large group size.
- Which particular variables are to be grouped together and which variables are to be put in different groups?? Naturally, the variables, that interact with each other i.e. that appear in a non-separable form in the objective function, demands to be grouped together in order to being optimized simultaneously.

Thus, proper selection of subcomponent size and appropriate grouping of variables is really a challenge for researchers in this domain. Various dynamically varying subgroup size selection have already been experimented. Also random grouping technique is invented, increasing the probability that the inter-dependent variables be optimized simultaneously. We have chosen a very good strategy to overcome these two difficulties too, which is discussed in an elaborated fashion in section VI. Right now, the modified DE variant chosen to embed the CC concept is demonstrated below.

IV. Subcomponent Optimizer Algorithm

In this section, we have briefly discussed Differential Evolution algorithm along with three new algorithmic components which have been introduced for more complex objective functions and thus, the algorithm is suitable as sub-optimizer for our aimed task. The three new components can be described as below

- First, to overcome the problem of being trapped in local optima, we proposed a control mutation strategy. the main aim of this strategy is to use the two classical DE mutation strategies- "DE/rand/1" and "DE/target-to-best/1". These two mutation strategies are chosen according to a probabilistic parameter or control parameter "*C*", which is decreasing linearly with iteration from an upper value to a lower value.
- Second, the perturbing vectors in mutation are chosen randomly using two different strategies- the first one in difference term are chosen applying selective pressure proposed by Whitely[10] and the second one is chosen randomly from the worst $p\%$ vectors of each generation (according to their fitness values). this is done with an intention to maintain diversity and exploration capability of the algorithm through out the search space.
- Third, the scaling factor(F) between the two essential parameters of classical DE algorithm for each individual target population vector is generated randomly using Cauchy distribution. This technique also maintains exploration capability of the algorithm due to the long-tail nature of Cauchy distribution. A pool of successful scaling factors, which were able to generate better trial vectors than the target vectors in current generation, are created. The mean of this pool of successful scaling factors and the location parameter (scaling parameter) of the Cauchy distribution of the current generation are combined according to a linear equation in order to create the next generation location parameter (scaling parameter) of the Cauchy distribution. The Crossover Probability, *CR* is kept constant.

This algorithm is termed as ADE_CM. For more detailed description, readers may go through the article [9].

V. High Frequency Random Grouping Coupled With Iteration Dependent Group Size Selection Scheme

This section is basically devoted for the techniques adopted by us to tackle the difficulties of the Cooperative Coevolution concept.

Yang *et al.* have already proved that random grouping of variables at the beginning of each cycle increases the probability of keeping two interacting variables in the same subgroup. In CCPSO2[13], the authors show that the probability of optimizing two interacting variables in the same subgroup for

at least one generation follows a binomial distribution and calculated as 99.48 %. Omidvar *et al.*[41] generalizes the probability calculation for at least k generations. They suggested that frequency of random grouping should be increased by a considerable amount to have these probabilities greater. In order to maximize the frequency of random grouping the sub-component optimizer should run for only one iteration, implying NP number of fitness evaluations per iteration. However, we have shown mathematically that the more frequent random grouping strategy adopted by Omidvar *et al.* really does not increase the probability of interacting variables to be optimized in the same group. Still, we prefer this scheme to the original one. The reason is followed by the mathematical analysis.

Mathematical Analysis

Under this subsection, we have calculated the probability of assigning v interacting variables. (say $x_1,\ x_2,\ ,...,\ x_v$) into one sub-component for at least k cycles for both cases : − the original random grouping scheme and the high frequency random grouping scheme separately with same number of FEs for both. The following notations are used for mathematical analysis :

$max\_FES\ =\ F$,
Total number of interacting variables = v,
Number of sub-components = m (total number of variables or group size),
Population size = NP.

Case I : Original Random Grouping Scheme

In this case, each sub-component should be optimized in each iteration and therefore $max\_iter(N_1)$ should have the following mathematical expression :

$$max\_iter(N_1)\ =\ \frac{F}{m * NP}, \tag{1}$$

as at each iteration, $(m*NP)$ numbers of FEs are evaluated. Now at any iteration, the total ways of orientation of v interacting variables into m subgroups are m^v as each variables can be assigned to any sub-group, having m possibilities and the total ways of keeping v interacting variables into one subgroup are m, as the variables together can be assigned to any one of the m subgroups.

Hence, the probability of grouping them in the same subgroup at any iteration is,

$$p\ =\ \frac{m}{m^v}\ =\ \frac{1}{m^{v-1}}, \tag{2}$$

So, the probability of assigning v interacting variables into one subcomponent for exactly r times out of N_1 times is:

$$P(X\ =\ r)\ =\ \binom{N_1}{r}.p^r.(1-p)^{(N_1-r)}, \tag{3}$$

where, $p\ =\ \frac{1}{m^{v-1}}$. Here, X is a random variable, signifying the number of times, the v interacting variables are grouped together. obviously, the maximum values of X is equal to N_1 and minimum value is $zero$. It follows a binomial probability distribution. Thus, the probability of grouping those v interacting variables into one sub-component at-least for k times would be,

$$P(X\ \geq\ r)\ =\sum_{r=k}^{N_1}\ (\frac{1}{m^{v-1}})^r.(1-\frac{1}{m^{v-1}})^{N_1-r} \tag{4}$$

Case II : High Frequency Random Grouing

In this case, at each iteration after random grouping only one sub-component is to be optimized (we have chosen the first sub-component randomly). So, $max\_iter\ (N_2)$ should have the the following mathematical expression :

$$max\_iter(N_2)\ =\ \frac{F}{NP}\ =\ m.NP, \tag{5}$$

as at each iteration, only NP number of FEs are evaluated. Now, also in this situation, total ways of orientation of v interacting variables into m subgroups are m^v.

But there is only one possibility to keep the v-interacting variables into the first subgroup. Hence, the possibility of grouping them in the first subgroup at any iteration is

$$p = \frac{1}{m^v} \tag{6}$$

So, the probability of assigning those v interacting variables to the first sub-group for exactly r times out of N_2 times is,

$$P(Y = r) = \binom{N_2}{r}.p^r.(1-p)^{(N_2-r)}. \tag{7}$$

Here, Y is the random variable, signifying the numbers of times the v interacting variables are grouped to the first subgroup. Naturally, Y follows a binomial distribution again.

Finally, the probability to group those v interacting variables in the first subgroup for at least k times would be,

$$P(Y\ \geq\ k)\ =\sum_{r=k}^{N_2}\binom{N_2}{r}.(\frac{1}{m^v})^r.(1-\frac{1}{m^v})^{N_2-r} \tag{8}$$

So, equation (6) and (10) are the final two equations, regarding our requirement.

Now, we are at the stage of comparing these two equations. If we take k=1, then from case 1,

$$P(X \geq 1) = 1 - P(X = 0) = 1 - (1 - \frac{1}{m^{v-1}})^{N_1}$$

and from case 2,

$$P(Y \geq 1) = 1 - P(Y = 0) = 1 - (1 - \frac{1}{m^v})^{N_2}.$$

Since, $N_2 = m.N_1$ so,

$$P(Y \geq 1) = 1 - P(Y = 0) = 1 - (1 - \frac{1}{m^v})^{N_2}$$

$$= 1 - (1 - \frac{1}{m^v})^{m.N_1}$$

. As, $\frac{1}{m^{v-1}} \ll 1$ and $\frac{1}{m^v} \ll 1$, we can do the following simplification,

$$1 - (1 - \frac{1}{m^{v-1}})^{N_1} \approx 1 - \frac{N_1}{m^{v-1}}$$

and,

$$(1-\frac{1}{m^v})^{m.N_1} \approx 1-\frac{m.N_1}{m^v}$$
$$= 1-\frac{N_1}{m^{v-1}}.$$

So, we can deduce that,

$$P(X \geq 1) \approx P(Y \geq 1).$$

Although, $\frac{1}{m^v}$ and $\frac{1}{m^{v-1}}$ are not always $\ll 1$, these two probabilities do not differ much from each other as shown below.

We have taken $D = 1000,\ F = 5000*D = 5e{+}06,\ NP = 100,\ v = 4$.

group size	m	N_1	N_2	$P(X \geq 1)$	$P(Y \geq 1)$
10	100	500	50000	0.00049	0.00049
20	50	1000	50000	0.0079	0.0079
50	20	2500	50000	0.2683	0.2683

Thus the above mathematical analysis clearly reveals that more frequent random grouping method does not improve the probability of interacting variables to be optimized together. In spite of that, we prefer this scheme due to its simplicity in coding. Also, in this case, the value of max_iter can be easily derived from max_FEs, just by dividing it by *NP*. Whereas, the original scheme involves complex calculation of max_iter. Even, in case of some of variable group size strategies, it is impossible to precisely determine the value of max_iter. The sub-component optimizer in our paper uses a mutation controlling parameter *C*, which is dependent on max_iter. So it is very important to calculate precisely the value of it. However, we can replace *(iter/*max_iter*)* by *(FEs/*max_FEs*)* in the expression of *C*, as these ratios are same. But the parameter will lose the property of uniform decreasing with iteration in that case due to the sub-component selection method used here(discussed below). Moreover, the probability calculations, shown in the tables, explore the fact that in both the cases the probabilities of interacting variables to be optimized in the same group are nearly same.

In a nutshell, high frequency random grouping simplifies the coding scheme and calculation of max_iter without degrading the performance of the algorithm in terms of probability thereby compelling us to go for high frequency random grouping.

Now the time is to discuss about subgroup size selection scheme. In MLCC, the authors used a complex formula which automatically adopts the proper group size from a predefined set of group sizes according to the landscape of the problem at hand. Yao *et al.* have experimented with another simple scheme in DECC-ML and also in CCPSO2. They preserved the predefined set of sub-component size of MLCC, but instead of the sophisticated formula, they choose the group size using uniform random number. If the chosen group size provides improvement, it is kept unchanged; otherwise, a new group size is selected from the same set using again a uniform random number and the above process continues.

We have proposed a new strategy in this regard. At the early stages of the generation, small group size is useful to locate good regions of the landscape quickly; and as it proceeds towards end, large group size, containing more global information, is needed for fine tuning. Keeping in mind the above stated concept, we formulated our methodology. Here also we have considered a pre-defined set of possible subgroup sizes, $S = [10, 20, 50]$.

Two transition parameters $TR1$ and $TR2$ are chosen to control the probabilities of selection of group size. These two parameters are varying with iteration as following:

$$TR1 = 0.6 - 0.5\frac{iter}{max\_iter}, \tag{9}$$

$$TR2 = 0.9 - 0.5\frac{iter}{max\_iter}. \tag{10}$$

The pseudo code for the group size selection is stated below and described elaborately followed by:

Algorithm 2 : Pseudo Code Of Group Size Selection

```
For i = 1 to NP
    TR = rand(0,1);
    if (TR <= TR1)
        s = S(1);
    else if (TR >= TR2)
        s = S(3);
    else
        s = S(2);
    end if
end for
```

Here, s represents group size for each population vector at each iteration. It is obvious from the coding scheme that at each iteration:

- The probability of selection of $S(1) = 10$ is: $TR1$.
- The probability of selection of $S(2) = 20$ is: $(TR2 - TR1)$.
- The probability of selection of $S(3) = 50$ is: $(1 - TR2)$

Now, according to the expressions of $TR1$ and $TR2$, they are linearly decreasing with iteration. $TR1$ is varying between 0.6 and 0.1 and$TR2$ is varying between 0.9 and 0.4. But at each stage, the difference $(TR2 - TR1)$ is constant $(= 0.3)$.Thus, it can be stated that probabilities of group size $10, 20$ and 50 are decreasing, constant and increasing respectively with iteration. Thus, at each generation a population vector selects the group size as per the above mentioned code. Following, the variables are permuted randomly and finally the first group size number of variables is optimized using the DE variant, stated above. So, high frequency random grouping is maintained and also as the algorithm proceeds towards the end, large group size is selected by more number of population vectors, giving more global information. We have shown statistically (in section VII) that the proposed scheme for sub-component size selection is better than that of DECC-ML, embedding their scheme with our algorithm.Right now, we are moving towards the new mechanism, we developed, inspired from human psychology, termed as perfunctory mechanism.

VI. Human Psychology motivated Perfunctory Mechanism

If the human behaviour is studied, it will be found that within a group of people, always there will be few many who are not rigid towards their duty. They do not follow strictly what they are told to do. This mentality can be termed as perfunctory. So, whenever a group of people are bound together to finish one particular task, it is expected that some of them possess this kind of mentality. Some of them are expected not to be firm about their responsibility. This particular perfunctory mentality is adopted by us.

In this case the objective is to find the minimum of a high dimensional function (the lowest valley of the landscape). For this task to be performed, NP numbers of people are hired from different regions. Differential Evolution mechanism along with Cooperative Coevolution is decided to be followed by this group of people to solve the problem at hand. So, it is expected that some of these NP numbers of people are not strict in their principle; not rigid in their duty. They have a mentality to deviate a little from the orders imposed on them. We have used this perfunctory mechanism in our algorithm as follows:

Algorithm 3 : Pseudo Code Of Perfunctory Mechanism

```
PR = 0.05 + 0.05 * rand;
if(rand <= PR)
    for q = 1 to n;
        r(q) = floor(4 * rand(0,1));
    end for
    BW = (ub - lb) * 0.1
    for q = 1 to n
        if(r(q) = = 0)
            V(q) = V(q) + BW * (-1 + rand * 2);
        end if
    end for
end if
```

Here, PR represents perfunctory rate, which varies between 0.05 and 0.1 randomly. PR actually decides how many of those NP people will follow this mechanism at every generation. At each generation, this 5% to 10% population deviate from their order, imposed on them by Differential Evolution algorithm. DE strategy gives a donor vector to each population vector and ordered them to make a crossover with these donors. But these small percentages of the whole population are not so rigid, preferring a slight deviation from the donors devoted to them.Thus nearly one fourth of the variables of the corresponding donors are changed according to the equation:

$$V(q) = V(q) + BW * (-1 + rand * 2) \tag{11}$$

BW, here, indicates the range of deviation i.e. bandwidth. We have chosen BW as the 10% of the variables' original range,

$$BW = (ub - lb) * 0.1. \tag{12}$$

This perfunctory mechanism makes the algorithm more realistic, reflecting the human behaviour properly. It may help the population to find good directions of search in the neighbourhood regions of donor vector points.It actually introduces some sort of exploration in the neighbourhood of donor, which is really very useful in this high dimensional search space. We found this scheme effective in high dimensional benchmark functions. So it is expected that this perfunctory mechanism will perform well on the real world problems.

VII. Experimental Settings

A. Benchmark Functions

To evaluate the performance of CCDE_PM algorithm, we take two benchmark suites proposed for the competitions on large scale single-objective global optimization with bound constraints, under the IEEE CEC 2008 and 2010 conferences[11],[12]. CEC 2008 benchmark suite consists of 7 functions in which the global optimum is shifted to a different value in each dimensions. The functions are briefly introduced below:

Separable Functions:

f_1: Shifted Sphere Function
f_4: Shifted Rastrigins Function
f_6: Shifted Ackleys Function

Non-separable Functions:

f_2: Shifted Schwefels Problem 2.21
f_3: Shifted Rosenbrocks Function
f_5: Shifted Griewanks Function
f_5: Fast Fractal DoubleDip Function

On the other hand, CEC 2010 contains 20 functions which are shifted and rotated in nature. Incorporation of rotation enhances further variable interactions making them non-separable. This test-bed is comprised with four categories of large scale problems:

Separable Functions:

f_1: Shifted Elliptic Function
f_2: Shifted Rastrigins Function
f_3: Shifted Ackleys Function

Partially-separable functions, having small number of dependent variables and the remaining independent variables (m=50):

f_4: Single-group Shifted and m-rotated Elliptic Function
f_5: Single-group Shifted and m-rotated Rastrigins Function
f_6: Single-group Shifted and m-rotated Ackleys Function
f_7: Single-group Shifted and m-dimensional Schwefels Problem 1.2.
f_8: Single-group Shifted and m-rotated Rosenbrocks Function

Partially-separable functions, consisting of multiple independent subcomponents, each of which is m-nonseparable:

f_9: D/2m- group shifted and m-rotated elliptic Function
f_{10}: D/2m- group shifted and m-rotated Rastrigins Function
f_{11}: D/2m- group shifted and m-rotated Ackleys Function
f_{12}: D/2m- group shifted and m-dimensional Schwefels Problem 1.2
f_{13}: /2m- group shifted and m-dimensional Rosenbergs Function
f_{14}: D/m- group shifted and m-rotated elliptic Function
f_{15}: D/2m- group shifted and m-rotated Rastrigins Function
f_{16}: D/2m- group shifted and m-rotated Ackleys Function
f_{17}: D/m- group shifted and m-dimensional Schwefels Problem 1.2
f_{18}: D/2m- group shifted and m-dimensional Rosenbergs Function

Fully Non-separable Functions:

f_{19}: Shifted Schwefels Problem 1.2
f_{20}: Shifted Rosenbergs Function

B. Algorithms Compared

The performance of CCDE_PM is compared with the algorithms like CCPSO2[13], sep-CMA-ES[14], DE with Self-Adaption and cooperative co-evolution (DEwSAcc) [21], Efficient Population Utilization Strategy for PSO (EPUS-PSO)[45], Multi-level Cooperative Coevolution (MLCC)[] on the CEC 2008 benchmark problems. EPUS-PSO[45] applies an adaptive control of the swarm size according to the search results. MLCC utilises CC approach (like CCPSO2) employing both random grouping and adaptive weighting, but they uses a sophisticated formula for group size selection, depending on the performance. DEwSAcc, MLCC and EPUS-PSO took part in the CEC 2008 competition on large scale global optimization.

On CEC 2010 benchmarks, CCDE_PM is statistically compared with the following algorithms: Memetic Algorithm Based on Local Search Chains (MA-SW-Chains)[**?**], Dynamic Multi-Swarm Particle Swarm Optimizer with Sub-regional Harmony Search (DMS-PSO-SHS)[42], Differential Ant-Stigmergy Algorithm (DASA)[36], Self-adaptive DE for large scale global optimization (jDElsgo)[44], Sequential DE Enhanced by Neighbourhood Search (SDENS)[35], Two-stage based Ensemble Optimization (EOEA)[43], MLCC[15], DECC-ML[41]. Results for the compared algorithms are taken from the respective papers and a homogeneous experimental environment is maintained in each case. The parametric set up of our algorithm has already been specified in the previous portions of the paper at the time of explanation of different strategies used. However, we have set $NP = 100$ for all dimensions $(100, 500, 1000 \ \& \ 2000)$.

TABLE I. EFFECTIVENESS OF GROUP SIZE SELECTION SCHEME - COMPARISON WITH RANDOM SELECTION METHOD OF DECC-ML USING CEC 2008 BENCHMARKS (500*D*)

Functions	Iteration Dependent Selection Method	Random Selection Method
f_1	**2.81e-13(8.34e-14)**	4.28e-13(5.69e-14)
f_2	**2.93e+01(9.11e+00)**	2.12e+02(9.23e+01)
f_3	**4.91e+02(3.31e+01)**	9.98e+02(7.73e+02)
f_4	**1.67e+02(6.23e+01)**	9.29e+02(6.91e+02)
f_5	**1.42e-13(0.00e+00)**	1.10e-12(3.11e-13)
f_6	3.97e-13(1.11e-14)	**3.71e-13(9.12e-14)**
f_7	**-7.92e+03(2.13e+00)**	-7.91e+03(3.88e+01)

TABLE II. EFFECTIVENESS OF PERFUNCTORY MECHANISM USING CEC 2008 BENCHMARKS (1000*D*)

Functions	With	Without
f_1	**4.23e-13(9.51e-14)**	4.32e-13(9.92e-14)
f_2	**4.39e+01(2.29e+01)**	9.56e+01(1.92e+01)
f_3	**8.11e+02(6.45e+00)**	1.49e+03(2.65e+00)
f_4	**5.52e+02(5.13e+02)**	1.00e+03(3.32e+02)
f_5	**3.22e-13(9.15e-14)**	3.27e-13(9.22e-14)
f_6	8.82e-13(9.83e-14)	**6.72e-13(8.92e-14)**
f_7	**-2.11e+04(8.29e+01)**	-1.54e+04(9.97e+01)

C. Simulation Strategy

The algorithm is run on total 27 functions as per the instruction indicated in [11] and [12]. For CEC 2008 functions experiments are carried out with $100, 500$, and 1000 and also in 2000 dimensions. Whereas, for CEC 2010 benchmarks only 1000 dimension is considered in order to save time as well as space.

For each test function, the average results of 25 independent runs and the corresponding standard deviations are tabulated. In case of CEC 2008 benchmark functions, maximum number of FEs is set to be $5000 * D$ (D stands for dimension of the function) and for CEC 2010 problems it is equal to $3000 * D$ following [11] and [12] respectively. To judge the quality of the results, a two-tailed t-test is conducted with a null hypothesis stating that there is no significant difference between the two algorithms in comparison. The null hypothesis is discarded if the p-value falls below the significance level $a = 0.05$. Best results are marked in bold throughout the result tables and the p-values regarding the t-test between the best and the second best performing algorithms are recorded in the list.

VIII. EXPERIMENTAL RESULTS AND ANALYSIS

This section constitutes with the experimental evidences to justify the importance of different sub-components of CCDE_PM algorithms followed by a detailed analysis of the comparison tables on CEC 2008 and 2010 benchmark problems.

A. Effectiveness of Different algorithmic Variants of CCDE_PM

Under this sub heading, we have considered the discussion of the significance of the group size selection scheme and the

newly introduced perfunctory mechanism scheme.

1) Effectiveness of Group Size Selection Strategy: In order to prove the effectiveness of the strategy proposed in CCDE_PM, we have tabulated statistical results of this scheme in table I, along with CCDE_PM with random group size selection strategy instead of the iteration dependent one used in our case. The results of table I are based on CEC 2008 benchmarks on 500 dimensions as shown there. If we observe table I, it can be noticed that the iteration based strategy outperforms the random one in most of the cases. Thus, the experimental results clearly reveal the effectiveness of the iteration dependent scheme.

2) Effectiveness of Perfunctory Mechanism: Table II represents the effectiveness of the newly introduced strategy perfunctory mechanism. Here, we have shown the results of the algorithms on CEC 2008 benchmarks on 1000 dimensions with and without perfunctory mechanism. It is observed that for the functions f_2, f_3 and f_4 the results of the algorithm with perfunctory mechanism are considerably better than the other. And for the other functions, the result is nearly same for both cases. Thus, it can be concluded that perfunctory method is really useful in large scale problems.

B. Comparison on CEC 2008 Benchmarks

In Tables III, IV and V, we have reported the results of comparing CCDE_PM with five other powerful high dimensional optimizers on seven benchmark functions from CEC 2008 competition in $100, 500$ and 1000 dimensions respectively. The results are tabulated in terms of mean and standard deviation of the best-of-the-run objective values recorded over 25 runs for each function, as stated earlier. The resulting values of error for other functions are taken from the respective literatures as indicated in the tabulation. Table VI is devoted for the ranking of algorithms with respect to the error values of the Tables III, IV and V. The average rank is shown in the rightmost columns of Table VI, VII and VIII which clearly reveal the superiority of CCDE_PM. It is interesting to note that, the average rank of CCDE_PM decreases as the dimension increases. For the functions f_2, f_5, f_6 and f_7, our algorithm outperforms all others. In case of f_1 and f_3, CCDE_PM ranked two in 100 and 500 dimensions. In thousand dimensions for f_1 the rank remains unchanged but for f_3, the rank switches to one. However, for f_4 the result of CCDE_PM is not up to the mark. Still, it can be stated that the overall results of CCDE_PM is significantly better than the other algorithms in the table.

.

C. Comparison on CEC 2010 Benchmarks

Tables IX is contributed to the high dimensional benchmarks of CEC 2010. Here, we only considered 1000 dimensions to statistically compare the performance of CCDE_PM with other algorithms. We have recorded the means and standard deviations of the best-of-the-run objective values over 25 independent runs, same as before, and the results for other algorithms are imitated from the respective literatures as indicated in the tables. Individual ranks of the algorithms are tabulated in Table X along with the average ranks in the right most column. It is observed that CCDE_PM statistically becomes winner securing lowest average rank in the table. It outperforms the other algorithms in 13 out of 20 functions, as noticed in the tabulated errors. So, obviously a close scrutiny declares the superiority of CCDE_PM over the others.

D. Comparison on $2000D$ Benchmarks

We further expanded the dimensionality of the problems to examine whether CCDE_PM is able to maintain a quality performance. We tested the algorithm on the CEC 2008 benchmark suites for 2000 dimensions and compared the results with those of CCPSO2 and sep-CMA-ES. However, we have not considered all the 7 functions of CEC 2008 but the first four ones for our comparison. Table X reflects the resulting comparison. We have seen that in 2000 dimension also CCDE_PM maintains the same quality and follows the same trend of rank. For f_1 and f_3, it beats CCPSO2 but fails toovertake sep-CMA-ES. In case of f_4, it is able to outperform sep-CMA-ES but remains inferior to CCPSO2. Finally for f_2, CCDE_PM secured the first rank, statistically outperforming the others.

IX. Conclusion

In this article, we have presented a Differential Evolution variant equipped with techniques to tackle high dimensional global optimization. We have embedded Cooperative Coevolution method on to the main DE framework for handling those problems. We removed the difficulties of Cooperative Coevolution efficiently. First of all, we have considered high frequency random grouping. It is shown mathematically that high frequency random grouping does not increase the probability of optimizing interacting variables into the same subgroup effectively. Still, we go for the strategy due to some reasons, discussed in detail inside the paper. We have used a very interesting scheme dependent on iteration for the group size selection. In this scheme, the probability of selection of group size, being large, is becoming higher as the iteration increases. Finally, we proposed a new strategy, inspired from human behaviour, making the algorithm much more realistic. We have termed it as perfunctory mechanism, which is also described elaborately inside the paper under the sub-heading human psychology motivated perfunctory mechanism. We have shown the effectiveness of the group size selection scheme and also the perfunctory mechanism, giving statistical results with and without those schemes. The results of our algorithm for CEC 2008 benchmarks are compared with 5 other state-of-the-art algorithms. And the results for CEC 2010 benchmark functions are compared with 9 popular algorithms. Both the tabulations explore the superiority of our algorithm, named as CCDE_PM, over the others. We have also presented a rank table for each benchmark suites, offering a better insight into the comparison tables. We have also run CCDE_PM on CEC 2008 for 2000 dimensions, where it is also able to stick to quality performance.

TABLE III. RESULTS OF CCDE_PM AND COMPETITOR ALGORITHMS ON CEC 2008 BENCHMARKS IN $100D$

Functions	f_1	f_2	f_3	f_4	f_5	f_6	f_7
CCPSO2	7.73e-14	6.08e+00	4.23e+02	3.98e-02	3.45e-03	1.44e-13	-1.50e+03
	(3.23e-14)	(7.83e+00)	(8.65e+02)	(1.99e-01)	(4.88e-03)	(3.06e-14)	(1.04e+01)
Sep-CMA-ES	**9.02e-15**	2.31e+01	**4.31e+00**	2.78+02	2.96e-04	2.12e+01	-1.39e+03
	(5.53e-15)	(1.39e+01)	**(1.26e+01)**	(3.43e+01)	(1.48e-03)	(4.02e-01)	(2.64e+01)
EPUS-PSO	7.47e-01	1.86e+01	4.99e+03	4.71e+02	3.72e-01	2.06e+00	-8.55e+02
	(1.70e-01)	(2.26e+00)	(5.35e+03)	(5.94e+01)	(5.60e-02)	(4.40e-01)	(1.35e+01)
MLCC	6.82e-14	2.52e+01	1.49e+02	**4.38e-13**	3.41e-14	1.11e-13	-1.54e+03
	(2.32e-14)	(8.73e+00)	(5.72e+01)	**(9.21e-14)**	(1.16e-14)	(7.87e-15)	(2.53e+00)
DEwSACC	5.68e-14	8.25e+00	1.44e+02	4.37e+00	3.06e-14	1.12e-13	-1.36e+03
	(0.0e+00)	(5.33e+00)	(5.85e+01)	(7.65e+00)	(7.87e-15)	(1.53e-14)	(2.46e+01)
CCDE_PM	5.68e-14	**1.13e-01**	9.03e+01	1.98e+01	**2.84e-14**	**1.13e-14**	**-1.62e+03**
	(0.00e+00)	**(3.22e-02)**	(1.64e+01)	(8.00e+00)	**(0.00e+00)**	**(0.00e+00)**	**(9.23e+00)**
p-values							

TABLE IV. RESULTS OF CCDE_PM AND COMPETITOR ALGORITHMS ON CEC 2008 BENCHMARKS IN $500D$

Functions	f_1	f_2	f_3	f_4	f_5	f_6	f_7
CCPSO2	3.00e-13	5.79e+01	7.24e+02	3.98e-02	1.18e-03	5.34e-13	-7.23e+03
	(7.96e-14)	(4.21e+01)	(1.54e+02)	(1.99e-01)	(4.61e-03)	(8.61e-14)	(4.16e+01)
Sep-CMA-ES	**2.25e-14**	2.12e+02	**2.93e+02**	2.18e+03	7.88e-04	2.15e+01	-6.37e+03
	(6.10e-15)	(1.74e+01)	**(3.59e+01)**	(1.51e+02)	(2.82e-03)	(3.10e-01)	(7.59e+01)
EPUS-PSO	8.45e+01	4.35e+01	5.77e+04	3.49e+03	1.64e+00	6.64e+00	-3.51e+03
	(6.40e+00)	(5.51e-01)	(8.04e+03)	(1.12e+02)	(4.69e-02)	(4.49e-01)	(2.10e+01)
MLCC	4.29e-13	6.66e+01	9.24e+02	**1.79e-11**	2.12e-13	5.34e-13	-7.43e+03
	(3.31e-14)	(5.70e+00)	(1.73e+02)	**(6.31e-11)**	(2.48e-14)	(7.01e-14)	(8.03e+00)
DEwSACC	2.09e-09	7.57e+01	1.81e+03	3.64e+02	6.90e-04	4.80e-01	-5.74e+03
	(4.62e-09)	(3.05e+00)	(2.74e+02)	(5.24e+01)	(2.41e-03)	(5.74e-01)	(1.84e+02)
CCDE_PM	2.81e-13	**2.93e+01**	4.91e+02	1.67e+02	**1.42e-13**	**3.97e-13**	**-2.11e+04**
	(8.34e-14)	**(9.11e+00)**	(3.31e+01)	(6.23e+01)	**(0.00e+00)**	**(1.11e-14)**	**(8.29e+01)**
p-values							

TABLE V. RESULTS OF CCDE_PM AND COMPETITOR ALGORITHMS ON CEC 2008 BENCHMARKS IN $1000D$

Functions	f_1	f_2	f_3	f_4	f_5	f_6	f_7
CCPSO2	5.18e-13	7.82e+01	1.33e+03	1.99e-01	1.18e-03	1.02e-12	-1.43e+04
	(9.61e-14)	(4.25e+01)	(2.63e+02)	(4.06e-01)	(3.27e-03)	(1.68e-13)	(8.27e+01))
Sep-CMA-ES	**7.81e-15**	3.65e+02	9.10e+02	5.31e+03	3.94e-04	2.15e+01	-1.25e+04
	(1.52e-15)	(9.02e+00)	(4.54e+01)	(2.48e+02)	(1.97e-03)	(3.19e-01)	(9.36e+01)
EPUS-PSO	5.53e+02	4.66e+01	8.37e+05	7.58e+03	5.89e+00	1.89e+01	-6.62e+03
	(2.86e+01)	(4.00e-01)	(1.52e+05)	(1.51e+02)	(3.91e-01)	(2.49e+0)	(3.18e+01)
MLCC	8.46e-13	1.09e+02	1.80e+03	**1.37e-10**	4.18e-13	1.06e-12	-1.47e+04
	(5.00e-14)	(4.75e+00)	(1.58e+02)	**(3.37e-10)**	(2.78e-14)	(1.06e-12)	(1.51e+01)
DEwSACC	8.79e-03	9.61e+01	9.15e+03	1.82e+03	3.58e-03	2.30e+00	-1.06e+04
	(5.27e-03)	(1.82e+00)	(1.26e+03)	(1.38e+02)	(5.73e-03)	(2.98e-01)	(4.18e+02)
CCDE_PM	4.23e-13	**4.39e+01**	**8.11e+02**	5.52e+02	**3.22e-13**	**8.82e-13**	**-1.54e+04**
	(9.51e-14)	**(2.29e+01)**	**(6.45e+00)**	(5.13e+02)	**(9.15e-14)**	**(9.83e-14)**	**(9.97e+01)**
p-values							

TABLE VI. RANKS OF CCDE_PM AND COMPETITOR ALGORITHMS ON CEC 2008 BENCHMARKS IN $100D$

Functions	f_1	f_2	f_3	f_4	f_5	f_6	f_7	Average Rank
CCPSO2	5	2	5	2	5	4	3	3.714
Sep-CMA-ES	1	5	1	5	4	6	4	3.714
EPUS-PSO	6	4	6	6	6	5	6	5.571
MLCC	4	6	4	1	3	2	2	3.142
DEwSACC	2	3	3	3	2	3	5	3
CCDE_PM	2	1	2	4	1	1	1	1.714

TABLE VII. RANKS OF CCDE_PM AND COMPETITOR ALGORITHMS ON CEC 2008 BENCHMARKS IN $500D$

Functions	f_1	f_2	f_3	f_4	f_5	f_6	f_7	Average Rank
CCPSO2	3	3	3	2	5	3	3	3.142
Sep-CMA-ES	1	6	1	5	4	6	4	3.857
EPUS-PSO	6	2	6	6	6	5	6	5.285
MLCC	4	4	4	1	2	2	2	2.714
DEwSACC	5	5	5	4	3	4	5	4.428
CCDE_PM	2	1	2	3	1	1	1	1.571

TABLE VIII. RANKS OF CCDE_PM AND COMPETITOR ALGORITHMS ON CEC 2008 BENCHMARKS IN $1000D$

Functions	f_1	f_2	f_3	f_4	f_5	f_6	f_7	Average Rank
CCPSO2	3	3	3	2	4	2	3	2.857
Sep-CMA-ES	1	6	2	5	3	6	4	3.714
EPUS-PSO	6	2	6	6	6	5	6	5.285
MLCC	4	5	4	1	2	3	2	3
DEwSACC	5	4	5	4	5	4	5	4.571
CCDE_PM	2	1	1	3	1	1	1	1.428

In future, we are planning to apply this algorithm to some real world high dimensional problems. We are also interested to see the effect of perfunctory mechanism on the other domains of problems, may be with DE or other evolutionary algorithms like PSO, GA etc. We believe that this mechanism will also show its power on other kinds of problems. Moreover, we are thinking about some hybrid algorithms using micro and CC approach for solving large scale problems.

REFERENCES

[1] K. Tang, X. Yao, P. N. Suganthan, C. MacNish, Y. P. Chen, C. M. Chen, and Z. Yang, "Benchmark Functions for the CEC2008 Special Session and Competition on Large Scale Global Optimization," *Technical Report, Nature Inspired Computation and Applications Laboratory, USTC, China, http://nical.ustc.edu.cn/cec08ss.php*, 2007.

[2] K. Foli, T. Okabe, M. Olhofer, Y. Jin, and B. Sendhoff, "Optimization of Micro Heat Exchanger: CFD, Analytical Results and Multiobjective Evolutionary Algorithms," *Int. J. Heat Mass Transfer*, vol. 49, nos. 5-6, pp. 1090-1099, 2006.

[3] T. Sonoda, Y. Yamaguchi, T. Arima, M. Olhofer, B. Sendhoff, and H.-A. Schreiber, "Advanced High Turning Compressor Airfoils For Low Reynolds Number Condition, Part I: Design and Optimization," *J. Turbomach*., vol. 126, no. 3, pp. 350-359, 2004.

[4] C. Wang and J. Gao, "High-dimensional Waveform Inversion With Cooperative Coevolutionary Differential Evolution Algorithm," *IEEE Geoscience and Remote Sensing Letters*, Vol. 9, No. 2, pp. 297- 301, March, 2012.

[5] S. Das and P. N. Suganthan, "Problem Definitions and Evaluation Criteria for CEC 2011 Competition on Testing Evolutionary Algorithms on Real World Optimization Problems," Technical Report, Jadavpur University, India and Nanyang Technological University, Singapore, 2010.

[6] A. Vicini and D. Quagliarella, "Airfoil and Wing Design Through Hybrid Optimization Strategies," in *Proc. 16th Appl. Aerodynamics Conf.*, AIAA paper 98-2729. 1998, pp. 1-11.

[7] M. A. Potter and K. A. D. Jong, A Cooperative Coevolutionary Approach To Function Optimization, in *Proc. of the Third Conference on Parallel Problem Solving from Nature*, vol. 2, 1994, pp. 249-257.

[8] R. Storn and K. V. Price, Differential evolutionA Simple and Efficient Adaptive Scheme For Global Optimization Over Continuous Spaces, ICSI, Berkeley, CA, Tech. Rep. TR-95-012. [Online].Available: http://http.icsi.berkeley.edu/ storn/litera.html.

[9] S. B. Roy, M. Dan, P. Mitra,"Improving Adaptive Differential Evolution with Con- trolled Mutation Strategy," *SEMCCO 2012*, LNCS Vol. 7677, pp. 636-643, 2012.Online version available at:http://link.springer.com/chapter/10.1007/978-3-642-35380-2_74

[10] Andrew M. Sutton, M. Lunacek, L. D. Whitely, "Differential Evolution and Non-separability: Using Selective pressure to Focus Search," in GECCO, London, England, UK(2007).

[11] K. Tang, X. Yao, P. Suganthan, C. MacNish, Y. Chen, C. Chen, and Z. Yang, "Benchmark Functions For the CEC2008 Special Session and Competition on Large Scale Global Optimization," Nature Inspired Computat. Applicat. Lab., Univ. Sci. Technol. China, Hefei, China, Tech. Rep., 2007 [Online]. Available: http://nical.ustc.edu.cn/cec08ss.php.

[12] K. Tang, X. Li, P. Suganthan, Z. Yang, and T. Weise, "Benchmark Functions For the CEC2010 Special Session and Competition on Large Scale Global Optimization," Nature Inspired Computat. Applicat. Lab., Univ. Sci. Technol. China, Hefei, China, Tech. Rep., 2009 [Online]. Available: http://nical.ustc.edu.cn/cec10ss.php.

[13] X. Li and X. Yao, "Cooperatively Coevolving Particle Swarms For Large Scale Optimization," *IEEE Trans. on Evolutionary Computation*, Accepted, 2011.

[14] R. Ros and N. Hansen, "A Simple Modification In CMA-ES Achieving Linear Time and Space Complexity," in *Proc. PPSN X*, pp. 296305, Apr. 2008.

[15] Z. Yang, K. Tang, and X. Yao, "Multilevel Cooperative Coevolution For Large Scale Optimization," in *Proc. IEEE Congr. Evol. Comput*., pp. 1663- 1670, Jun. 2008.

[16] S. Z. Zhao, J. J. Liang, P. N. Suganthan, and M. F. Tasgetiren, "Dynamic Multi-swarm Particle Swarm Optimizer With Local Search For Large Scale Global Optimization," *IEEE Congress on Evolutionary Computation*(CEC 2008), pp. 3845 3852, Hong Kong, June 2008.

[17] Z. Yang, K. Tang, and X. Yao, "Large Scale Evolutionary Optimization Using Cooperative Coevolution," *Inform. Sci*., vol. 178, no. 15, pp. 2986-2999, Aug. 2008.

[18] Z. Yang, K. Tang, and X. Yao, "Differential Evolution For High Dimensional Function Optimization," in *Proc. IEEE Congr. Evol. Comput.*, Sep. 2007, pp. 3523-3530.

TABLE IX. RESULTS OF CCDE_PM AND COMPETITOR ALGORITHMS ON CEC 2010 BENCHMARKS IN $1000D$

Algo	jDElsgo	DECC-ML	DASA	DMS-PSO-SHS	SDENS	EOEA	DECC-CG	MLCC	MA-SW-Chains	CCDE_PM	p-values
Error	Mean Std dev.	Mean Std dev.	Mean Std dev.	Mean Std dev.	Mean Std dev.	Mean Std dev.	Mean Std dev.	Mean Std dev.	Mean Std dev.	Mean Std dev.	
f_1	8.86e-20 4.51e-20	1.93e-25 1.86e-25	1.52e-21 2.33e-21	5.51e-15 4.00e-14	5.73e-06 4.46e-06	2.20e-23 2.87e-23	2.93e-07 8.62e-08	1.53e-27 7.66e-27	2.10e-14 1.99e-14	**1.50e-27** **7.13e-28**	
f_2	1.25e-01 3.45e-01	2.17e+02 2.98e+01	8.48e+00 2.52e+00	8.51e+01 2.06e+01	2.21e+03 8.95e+01	3.62e-01 6.71e-01	1.31e+03 3.26e+01	5.57e-01 2.21e+00	8.10e+02 5.88e+01	**2.11e-02** **6.66e-03**	
f_3	3.81e-12 5.02e-12	**1.18e-13** **8.22e-15**	7.20e-11 8.27e-12	5.52e-11 3.25e-10	2.70e-05 1.54e-05	1.67e-13 1.13e-14	1.39e+00 9.73e-02	9.88e-13 3.70e-12	7.28e-13 3.40e-13	1.41e-13 2.15e-13	
f_4	8.06e+10 3.08e+10	3.58e+12 1.54e+12	5.05e+11 2.22e+11	2.46e+11 3.31e+10	5.11e+12 2.16e+12	3.09e+12 1.61e+12	1.70e+13 5.37e+12	9.61e+12 3.43e+12	3.53e+11 3.12e+10	**7.92e+10** **6.23e+10**	
f_5	9.72e+07 1.44e+07	2.99e+08 9.31e+07	6.20e+08 7.87e+07	8.36e+07 6.10e+06	1.18e+08 2.88e+07	**2.24e+07** **5.91e+06**	2.63e+08 8.44e+07	3.84e+08 6.93e+07	1.68e+08 1.04e+08	1.03e+08 2.71e+07	
f_6	**1.70e-08** **4.03e-08**	7.93e+05 3.97e+06	1.97e+07 4.45e+04	8.28e-02 9.96e-01	2.02e-04 4.29e-05	3.85e+06 4.97e+05	4.96e+06 8.02e+05	1.62e+07 4.97e+06	8.14e+04 2.84e+05	3.32e+00 1.11e+00	
f_7	1.31e-02 6.38e-02	1.39e+08 7.72e+07	7.78e+00 3.10e+00	1.95e+03 1.56e+02	1.20e+08 6.56e+07	1.24e+02 1.55e+02	1.63e+08 1.37e+08	6.89e+05 7.37e+05	1.03e+02 8.70e+01	**7.73e-03** **3.14e-04**	
f_8	3.15e+06 3.27e+06	3.46e+07 3.56e+07	4.98e+07 8.95e+07	1.29e+07 1.91e+06	5.12e+07 2.12e+07	1.01e+07 1.28e+07	6.44e+07 2.89e+07	4.38e+07 3.45e+07	1.41e+07 3.68e+07	**1.78e+06** **1.45e+06**	
f_9	3.11e+07 5.00e+06	5.92e+07 4.71e+06	3.60e+07 4.78e+06	**8.72e+06** **6.51e+05**	5.63e+08 5.78e+07	4.63e+07 4.78e+06	3.21e+08 3.38e+07	1.23e+08 1.33e+07	1.41e+07 1.15e+06	9.22e+06 7.14e+05	
f_{10}	2.64e+03 3.19e+02	1.25e+04 2.66e+02	7.29e+03 2.69e+02	5.53e+03 5.18e+02	6.87e+03 5.60e+02	**1.08e+03** **6.91e+01**	1.06e+04 2.95e+02	3.43e+03 8.72e+02	2.07e+03 1.44e+02	2.19e+03 2.83e+01	
f_{11}	2.20e+01 1.53e+01	**1.80e-13** **9.88e-15**	1.98e+02 1.57e-01	3.24e+01 3.00e+00	2.21e+02 5.09e-01	3.86e+01 1.65e+01	2.34e+01 1.78e+00	1.98e+02 6.98e-01	3.80e+01 7.35e+00	9.32e+00 2.48e+00	
f_{12}	1.21e+04 2.04e+03	3.80e+06 1.50e+05	1.78e+03 2.25e+02	6.12e+02 6.00e+01	4.13e+05 4.28e+04	1.37e+04 2.90e+03	8.93e+04 6.87e+03	3.49e+04 4.29e+03	3.62e-06 5.92e-07	**9.98e-07** **2.41e-07**	
f_{13}	7.11e+02 1.37e+02	1.14e+03 4.31e+02	1.21e+03 7.39e+02	1.22e+03 1.05e+02	2.19e+03 1.03e+03	1.24e+03 4.59e+02	5.12e+03 3.95e+03	2.08e+03 7.27e+02	1.25e+03 5.72e+02	**6.21e+02** **8.08e+01**	
f_{14}	1.69e+08 2.08e+07	1.89e+08 1.49e+07	1.00e+08 7.88e+06	1.75e+07 1.55e+06	1.88e+09 2.33e+08	1.65e+08 8.95e+06	8.08e+08 6.07e+07	3.16e+08 2.77e+07	3.11e+07 1.93e+06	**8.85e+06** **7.32e+06**	
f_{15}	5.84e+03 4.48e+02	1.54e+04 3.59e+02	1.45e+04 3.69e+02	4.08e+03 2.17e+02	7.32e+03 9.63e+01	2.14e+03 1.22e+02	1.22e+04 8.97e+02	7.11e+03 1.34e+03	2.74e+03 1.22e+02	**1.89e+03** **5.97e+02**	
f_{16}	1.44e+02 3.43e+01	**5.08e-02** **2.54e-01**	3.97e+02 2.18e-01	6.98e+01 4.24e+00	4.08e+02 2.53e+00	8.26e+01 1.68e+01	7.66e+01 8.14e+00	3.76e+02 4.71e+01	9.98e+01 1.40e+01	2.98e+02 1.65e+01	
f_{17}	1.02e+05 1.26e+04	6.54e+06 4.63e+05	1.03e+04 8.99e+02	3.83e+03 4.15e+02	1.08e+06 1.11e+05	7.93e+04 8.80e+03	2.87e+05 1.98e+04	1.59e+05 1.43e+04	1.24e+00 1.25e-01	**1.04e-02** **9.31e-01**	
f_{18}	1.85e+03 3.18e+02	2.47e+03 1.18e+03	4.92e+03 2.28e+03	2.25e+03 1.16e+02	3.08e+04 1.22e+04	2.94e+03 6.92e+02	2.46e+04 1.05e+04	7.09e+03 4.77e+03	1.30e+03 4.36e+02	**1.26e+03** **1.34e+02**	
f_{19}	2.74e+05 2.12e+04	1.59e+07 1.72e+06	8.34e+05 5.26e+04	1.17e+06 1.06e+05	8.80e+05 1.59e+05	1.84e+06 9.97e+04	1.11e+06 5.15e+04	1.36e+06 7.35e+04	2.85e+05 1.78e+04	**8.21e+04** **7.35e+03**	
f_{20}	1.53e+03 1.32e+02	9.91e+02 3.51e+01	1.13e+03 1.79e+02	3.51e+02 4.01e+01	9.90e+02 1.62e+01	1.97e+03 2.35e+02	4.06e+03 3.66e+02	2.05e+03 1.80e+02	1.07e+03 7.29e+01	**2.41e+02** **3.22e+01**	

[19] X. Li and X. Yao, "Tackling High Dimensional Nonseparable Optimization Problems By Cooperatively Coevolving Particle Swarms," in *Proc. IEEE CEC*, pp. 1546-1553, May 2009.

[20] M. Omidvar, X. Li, X. Yao, and Z. Yang, "Cooperative Co-evolution For Large Scale Optimization Through More Frequent Random Grouping," in *Proc. CEC*, Jul. 2010, pp. 1754-1761.

[21] A. Zamuda, J. Brest, B. Boskovic, and V. Zumer, "Large Scale Global Optimization Using Differential Evolution With Self-adaptation and Cooperative Co-evolution", *IEEE Congress on Evolutionary Computation*(CEC 2008), pp. 3718-3725, Hong Kong, June 2008.

[22] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, 1989.

[23] K. Krishnakumar, "Micro-Genetic Algorithms For Stationary And Non-stationary Function Optimization," SPIE 1196, *Intelligent Control and Adaptive Systems*, 1989.

[24] G. Alvarez, *Can We Make Genetic Algorithms Work in High Dimensionality Problems?* Stanford Exploration Project (SEP) report, 112, 2002.

[25] T. Huang and A. S. Mohan, "MicroParticle Swarm Optimizer For Solving High Dimensional Optimization Problems", *Applied Mathematics and Computation*, 181(2):1148-1154, 2006.

[26] N. Krasnogor and J. Smith, "A Tutorial For Competent Memetic Algorithms: Model, Taxonomy, and Design Issue", *IEEE Transactions on Evolutionary Computation*, 9(5):474-488, 2005.

[27] N. Noman and H. Iba, "Enhancing Differential Evolution Performance With Local Search For High Dimensional Function Optimization", *In*

TABLE X. RANKS OF CCDE_PM AND COMPETITOR ALGORITHMS ON CEC 2010 BENCHMARKS IN $1000D$

Algorithm	jDElsgo	DECC-ML	DASA	DMS-PSO-SHS	SDENS	EOEA	DECC-CG	MLCC	MA-SW-Chains	CCDE_PM
f_1	6	3	5	7	10	4	9	2	8	1
f_2	2	7	5	6	10	3	9	4	8	1
f_3	6	1	8	7	9	3	10	5	4	2
f_4	2	7	5	3	8	6	10	9	4	1
f_5	3	8	10	2	4	1	7	9	6	5
f_6	1	6	10	3	2	7	8	9	5	4
f_7	2	9	3	6	8	5	10	7	4	1
f_8	2	6	8	4	9	3	10	7	5	1
f_9	4	7	5	1	10	6	9	8	3	2
f_{10}	4	10	8	6	7	1	9	5	2	3
f_{11}	3	1	8	5	10	7	4	9	6	2
f_{12}	5	10	4	3	9	6	8	7	2	1
f_{13}	2	3	4	5	9	6	10	8	7	1
f_{14}	6	7	4	2	10	5	9	8	3	1
f_{15}	5	10	9	4	7	2	8	6	3	1
f_{16}	6	1	9	2	10	4	3	8	5	7
f_{17}	6	10	4	3	9	5	8	7	2	1
f_{18}	3	5	7	4	10	6	9	8	2	1
f_{19}	2	10	4	7	5	9	6	8	3	1
f_{20}	7	4	6	2	3	8	19	9	5	1
Total	77	125	126	82	159	97	166	143	87	38
Average Rank	3.85	6.25	6.3	4.1	7.95	4.81	8.3	7.15	4.35	1.9

Proceedings of the conference on Genetic and evolutionary computation (GECCO '05), H.- G. Beyer (Ed.). ACM, New York, NY, USA, 967-974, 2005.

[28] S. Z. Zhao, P. N. Suganthan, and S. Das, "Self-adaptive Differential Evolution With Multi-Trajectory Search For Large Scale Optimization," *Soft Computing*, Vol. 15, pp. 2175-2185, 2011.

[29] D. Molina, M. Lozano, and F. Herrera, "MA-SW-Chains: Memetic Algorithm Based on Local Search Chains For Large Scale Continuous Global Optimization," *IEEE Congress on Evolutionary Computation*(CEC 2010), pp.3153-3160, Barcelona, July, 2010.

[30] D. Molina, M. Lozano, A. M. Snchez, and F. Herrera, "Memetic Algorithms Based On Local Search Chains For Large Scale Continuous Optimization Problems: MA-SSW-Chains", *Soft Computing*, 15, pp. 2201-2220, 2011.

[31] J. Brest, Zamuda, B. Boskovic, M. S. Maucec, and V. Zumer, "High Dimensional Real-Parameter Optimization Using Self-adaptive Differential Evolution Algorithm With Population Size Reduction," *IEEE Congress on Evolutionary Computation*(CEC 2008), pp.2032 - 2039, Hong Kong, June 2008.

[32] S.-T. Hsieh, T.-Y. Sun, C.-C. Liu and S.-J. Tsai, "Solving Large Scale Global Optimization Using Improved Particle Swarm Optimizer," *IEEE Congress on Evolutionary Computation*(CEC 2008), pp. 1777 - 1784, Hong Kong, June 2008.

[33] S. Rahnamayan and G. G. Wang, "Solving Large Scale Optimization Problems By Opposition-Based Differential Evolution (ODE)," World Scientific and Engineering Academy and Society, in *Transactions on Computers*, Volume 7, Issue 10, pp. 17921804, 2008.

[34] H. Wang, Z. Wu, S. Rahnamayan, and L. Kang, "A Scalability Test For Accelerated DE Using Generalized Opposition-Based Learning," in *Proc. Intelligent System Design and Applications*, pp. 1090-1095, 2009.

[35] H. Wang, Z. Wu, S. Rahnamayan and D. Jiang, "Sequential DE Enhanced by Neighborhood Search for Large Scale Global Optimization," *IEEE Congress on Evolutionary Computation*(CEC 2010), pp. 4056-4062, Barcelona, July, 2010.

[36] P. Korosec, K. Tashkova, and J. Silc, "The Differential Ant-Stigmergy Algorithm For Large-Scale Global Optimization," *IEEE Congress on Evolutionary Computation*(CEC 2010), pp. 4288-4295, Barcelona, July, 2010.

[37] M. Potter, "The Design and Analysis of a Computational Model of Cooperative Coevolution," Ph.D. dissertation, George Mason University, 1997.

[38] Y. Liu, X. Yao, Q. Zhao, and T. Higuchi, "Scaling up Fast Evolutionary Programming with Cooperative Coevolution," *Proceedings of the 2001 Congress on Evolutionary Computation*, pp. 1101-1108, 2001.

[39] Y. Shi, H. Teng, and Z. Li, "Cooperative Co-evolutionary Differential Evolution for Function Optimization," *Proceedings of the First International Conference on Natural Computation,* pp. 1080-1088, 2005.

[40] X. Cao, H. Qiao, and J. Keane, "A Low-Cost Pedestrian-Detection System With a Single Optical Camera," *IEEE Transactions on Intelligent Transportation Sytems*, vol. 8, no. 4, to appear, 2007.

[41] M. Omidvar, X. Li, X. Yao, and Z. Yang, Cooperative Co-evolution For Large Scale Optimization Through More Frequent Random Grouping, in Proc. CEC, Jul. 2010, pp. 1754-1761.

[42] S.-Z Zhao, P. N. Suganthan, and S. Das, "Dynamic Multi-swarm Particle Swarm Optimizer With Subregional Harmony Search," *IEEE Congress on Evolutionary Computation*(CEC 2010), pp. 1983-1990, Barcelona, July, 2010.

[43] Y. Wang and B. Li, "Two-Stage Based Ensemble Optimization For Large-Scale Global Optimization," *IEEE Congress on Evolutionary Computation*(CEC 2010), pp. 4488-4495, Barcelona, July, 2010.

[44] J. Brest, A. Zamuda, I. Fister, and M. S. Maucec, "Large Scale Global Optimization Using Self-Adaptive Differential Evolution Algorithm," *IEEE Congress on Evolutionary Computation*(CEC 2010), pp.3097-3104, Barcelona, July, 2010.

[45] S. Hsieh, T. Sun, C. Liu, and S. Tsai, "Solving Large Scale Global Optimization Using Improved Particle Swarm Optimizer," in *Proc. IEEE CEC*, pp. 1777-1784, Jun. 2008.