2013 IEEE Congress on Evolutionary Computation
June 20-23, Cancún, México

Micro-Differential Evolution with Local Search for High Dimensional Problems

Mauricio Olguín-Carbajal
Centro de Innovación y
Desarrollo Tecnológico en Cómputo
Instituto Politécnico Nacional
México D.F., México
Email: molguinc@ipn.mx

Enrique Alba
Universidad de Málaga, España
Departamento de Lenguajes y
Ciencias de la Computación
E.T.S.I. Informática (3-2-12)
Email: eat@lcc.uma.es

Javier Arellano-Verdejo
Centro de Investigación en Computación
Instituto Politécnico Nacional
México D.F., México
Email: jarellano@cic.ipn.mx

***Abstract*—Reduced population algorithms have proven to be efficient for solving optimization problems in the past. In this paper, we incorporate a local search procedure into a micro differential evolution algorithm (DE) with the aim of tackling high dimensional problems. Our main purpose is to find out if our proposal is more competitive in these problems than a canonical differential evolution algorithm. In relation to the state of the art techniques, the results our micro-DELS are comparable (or better) with the reference algorithms DECC-G and MLCC. This empirical analysis supports our conjecture that a reduced population DE hybridized with local search (our micro- DELS) is a key combination in dealing with functions having high dimensionality at a low computational cost.**

I. Introduction

Creating more efficient and accurate algorithms is a mainstream in optimization. On the one hand, many strategies have been developed with the aim of incrementing the efficiency of Evolutionary algorithms; one of these strategies is the use of reduced populations. On the other hand, hybridization is a well established line of research to improve the accuracy of a technique. In this work we propose a micro Differential Evolution algorithm (micro-DE) with a Local Search (LS) named micro-DELS, to improve efficiency and accuracy in high dimensional optimization problems.

The related literature shows many examples of reduced population algorithms for getting better results in Genetic Algorithms. A first approach was developed by Krishnakumar [4]. Krishnakumar used a population of five individuals and adopted an elitist strategy where he copied the best string found in the current population to the next generation. Krishnakumar reported better and faster results for its micro-GA with respect to a Genetic Algorithm [4] in two static functions and in two control real world problems. However, these approaches have been not tested in high dimensional problems.

Later, Huang and Mohan designed a Micro PSO [12] for high dimensional problems, where a small population has a better performance compared to a standard PSO. The superior performance was confirmed by the results of experiments with five functions (Rastrigrin, Rosenbrock, Griewank, Schwefel, and Ackley). All functions were tested with 500 dimensions, run with a limit of 3000 FE's (Function Evaluations), with a 30 particle population for PSO versus 3 particles for MPSO. The conclusion was that, in all the functions, MPSO outperformed PSO. However, the use of a more complex benchmark is needed to actually show the real impact of algorithms with a reduced population.

In a more recent work, Parsopoulos [8] proposes a Cooperative micro-Differential Evolution (COMDE) for high dimensional problems. His approach employs small cooperative subpopulations to detect subcomponents of the original problem solution concurrently. His experiments showed that COMDE produced (for all the tested operators) better results than a standard DE, for dimensions: 300, 600, 900, and 1200. However, his tests only used five functions for experiments: Sphere, Generalized Rosenbrock, Rastrigin, Griewank, and Ackley.

In summary, all previous works on micro-DE where very lightly evaluated on a few add-hoc functions, and not including a structured analysis on accuracy merging hybridization plus the canonical DE.

We have incorporated three mechanisms to the micro-DE with the aim of enhancing its efficiency. First, we design a *tracking-individual*, in order to follow the best individual by a short random distance for local exploration. Second, *restart-individual* for maintain diversity. Third, a *local search mechanism*. To evaluate the efficiency of the resulting approach, we have followed the experimental framework proposed in the Special Session for Large Scale Optimization Problems from CEC2012. We also compare the performance of micro-DELS against DECC-G, and MLCC the reference algorithms in special session LSGO of CEC2012.

Our results confirm that micro-EDLS is more efficient than prior micro-ED algorithms. We show that it is competitive in thirteen out of the twenty functions of the testbed used. Besides, by the use of a non-parametrical statistical test we make a ranking of the algorithm and validate our conclusions.

The remaining parts of the paper are as follows: Section 2 describes the DE and micro-DE algorithms basis as well as the local search procedure we use. Section 3 introduces our micro-DELS approach. Experiments set-up are described in Section 4 as well as the results are reported in Section 5. The analysis of the experiments results are presented in Section 6 and the paper concludes in Section 7.

II. Background

In this section we include some basic knowledge on DE, micro-DE, and the Local Search method we will use here.

First, let us define a large-scale optimization problem as follows: given an objective function $f(x)$ let be $\vec{x} = (x_1, ..., x_D)$ a vector of D variables in a decision space S. The variables $x_i \in \{x_{lb}, x_{ub}\}$ are limited by their lower $x_{lb} \in R$ and upper $x_{ub} \in R$ bounds. The goal of optimization is to find the minimum (or the maximum) of the objective function. When we refer to a large-scale optimization we talk about vectors of more than five hundred variables, i.e., $D \geq 500$.

A. Differential Evolution Algorithm (DE)

DE was initially created to deal with global optimization mainly in continuous spaces. Storn and Price defined DE as a *"...parallel direct search method which utilizes Np parameter vectors as a population for each G generation"* [9], so:

$$\vec{x}_{i,G} = 1, 2, ..., Np \tag{1}$$

where *Np* does not change during the algorithm execution. The initial population involves the generation of a set of vectors with initial random values. Each $\vec{x}_{i,G}$ individual of the initial population is evaluated to get its fitness value, and the result is assigned as the $\vec{x}_{i,G}$ aptitude. Our population is conformed of all vectors $\vec{x}_{i,G}$:

$$P = \{\vec{x}_1, ...\vec{x}_n\} \quad n \in [1, Np] \tag{2}$$

The DE population evolves by the use of two operators, known as *mutation* and *recombination*. These two operators produce new individuals. These new individuals are added to the present population, and then a selection step takes place performing an elitist construction of the new population.

The mutation operator represents the DE control procedure as a scheme for generating fresh individuals by adding a difference vector between two random population individuals, to a third individual. The random individuals are referred as r_1, r_2 and r_3, where the base vector is r_3. A special factor, called F, controls the scaling of the difference between vectors. This is, for each $\vec{x}_{i,G} = 0, 1, 2, ...,$ *Np*, a trial vector $\vec{v}$ is generated as follows:

$$\vec{v}_G = \vec{x}_{r_1,G} + F \times (\vec{x}_{r_2,G} - \vec{x}_{r_3,G}), \tag{3}$$

with $\quad I_k, r_1, r_2, r_3 \in [1, Np], \quad I_k \neq r_1 \neq r_2 \neq r_3$

where Ik is a population selected individual to test against trial vector.

Next, the recombination operator is applied on the $\vec{v}_G$ and the population vectors, producing trial vectors $\vec{u}_{i,G}$, for $i \in [1, Np]$. The vector

$$\vec{u} = (u_1, u_2, ..., u_D) \tag{4}$$

with

$$u_{ij,G+1} = \begin{cases} v_{ij,G+1}, & \text{if} \quad R_j \leq CR \quad or \quad j = r_k \\ x_{ij,G}, & \text{if} \quad R_j > CR \quad and \quad j \neq r_k \end{cases} \tag{5}$$

where *CR* $\in [0, 1]$ is used as a Combination Rate value parameter for the $\vec{u}_{i,G}$ trial vector generation; $j = 1, 2, ..., n$; R_j is a $j - th$ random of a uniform random number generator [0,1]. Sampled a new for every component of the vector $\vec{v}$. While $r_k \in [1, Np]$ is the random uniform individual index, and $\vec{x}_{i,G} \in [1, Np]$ is a population selected individual.

Finally, we compare the trial vector $\vec{u}_{i,G}$ with the actual $\vec{x}_{i,G}$ selected parent and the one with the better fitness passes to the next generation. The comparison of $\vec{u}_{i,G}$ versus $\vec{x}_{i,G}$ is done with all individuals of the current population, one by one. The individual with the better fitness (the minor value) passes to the next generation as eq. 6 indicates:

$$\vec{x}_{i,G+1} = \begin{cases} \vec{u}_{i,G}, & \text{if} \quad f(u) \leq f(x) \\ \vec{x}_{i,G}, & \text{if} \quad f(u) > f(x) \end{cases} \tag{6}$$

Storn and Price highlighted many variants of the DE algorithm, [9]. Such variants range from changing the number of parents to generate a test vector and the process of mutation involved. The different schemes to name DE variants are according to the general convention, DE/x/y/z where:

1) **DE** refers to a Differential Evolution Algorithm.
2) **x** denotes the mechanism to select a vector $\vec{X}_{r1}$.
3) **y** is the quantity of weighted difference vectors F $(\vec{X}_{r2} - \vec{X}_{r3})$ used to perturb $\vec{X}_{r1}$.
4) **z** refers to the crossover scheme used.

The most commonly used variant is $DE/rand/1/bin$ and is used for this test in all algorithms. This means we use a Differential Evolution (DE) with a random selected vector (rand) with one weighted difference vectors (1) and a binomial crossover scheme. For further reference check [11].

B. Micro Differential Evolution Algorithm (micro-DE)

A micro Differential Evolution algorithm (micro-DE) is a DE algorithm with a small population. Storn and Price recommend a population number of size up to $N = 10D$, were D is the problem dimension [9]. Due the small number of individuals, a micro-DE is expected to converge very fast, and get stuck on local optima. According to the relation $N = 10D$, when used in high dimensional problems, $D \geq 500$, the recommended population must be five thousand individuals and with $D \geq 1000$, the populations must be ten thousand individuals, according to the relation suggested by Storn and Price [9].

The ratio D/N is indicative, in most cases, of the difficulty met by an algorithm. As Parsopoulos indicates "*Empirical evidence suggest that in most cases the higher the ratio is, the harder the problem becomes for the algorithm*" [8]. However, we think that is possible to achieve better values than expected by using a population $N = 5$ and by adding a local search and a restart individual (as diversity mechanism).

From eq. 3 we can see that, in order to allow the application of the mutation operator, a DE algorithm must have at least

4 individuals. However on our tests, we find that the better results where achieved by using 5 individuals.

C. Local Search

There exists a huge variety of local search techniques for optimization. It is difficult to make an *a priory* selection of which one could actually help our micro-DE, but we take a look at the large scale continuous optimization procedures and we find the local search procedure of MTS.

The local search mechanism LS1 of the Multiple Trajectory Search algorithm (MTS), proposed by Tseng et al., [6], was initially designed for multi-objective optimization but also recently used in large scale optimization, where it showed the best performance in the CEC'08 special session on Large Scale Global Optimization competition [5]. MTS can be divided into two parts:

1) Generation of initial candidate solutions.
2) Local Search.

The first step is the generation of M initial solutions by using the Simulated Orthogonal Array (SOA), see [6].

The second step, in MTS, consists of iterations of several local search (LS) algorithms until the stop criterion is reached. At the first iteration, MTS makes local search all over the M initial solutions. For the rest of iterations MTS conducts local search over some selected good solutions. MTS is having three local search methods: LS1, LS2 and LS3. These three methods are applied to each individual and the best out of the three candidate solutions is selected for the next iteration until achieves the stop criterion.

Since every local search operation consumes $2 \times D$ fitness function evaluations, with D as the problem dimension, we here decided to use only one local search. LS1 is the local search that we selected because it achieved the best results (out of of the three LSs) in MTLS [6]. LS1 has also shown its advantages in hybridization when it was used in other works [3].

LS1 search along the solution vector from the first to the last dimension, as shown Algorithm 1. LS1 uses three initial parameters:

1) A boolean value *Improve*.
2) A Search Range *SR* value.
3) A solution to be optimized $\vec{x}^b$, in our case the best individual found by the micro-DE stage.

The *Improve* and *SR* values are initialized from lines 1 to 7 of Algorithm 1. After initialization step, begins the improvement of all the vector $\vec{x}^b$ components, starting with the variable x_i (with $i \leftarrow 1$) and assigning the value $x_i^b - SR$ (line 9). With this new x_i value we proceed to test the function aptitude (line 10). If there is any improvement with respect to the prior x_i^b value, the new x_i value is retained, *Improve* $\leftarrow TRUE$ and continue with the next x_i value (incrementing i value one unit). Otherwise the old x_i^b value is restored (line 14) and continue to the next part of the LS algorithm.

We assign into x_i (with $i \leftarrow 1$) the $x_i^b + 0.5 \times SR$ value (line 15) and test the function aptitude. If there is any improvement with respect to the previous x_i^b value we retain the new x_i value

Algorithm 1 Pseudocode for LS1

```
1: if $NOT\ Improve$ then
2:     $SR \leftarrow SR/2$
3:     if $SR < 3 \times 10^{-5}$ then
4:         $SR \leftarrow (MAX - MIN) \times 0.4$
5:     end if
6: end if
7: $Improve \leftarrow FALSE$
8: for $i = 1$ to $D$ do
9:     $x_i \leftarrow x_i^b - SR$
10:    if $f(x) \leq f(x^b)$ then
11:        $x_i^b \leftarrow x_i$
12:    end if
13:    if $f(x) > f(x^b)$ then
14:        $Restore(x_i)$
15:        $x_i \leftarrow x_i^b + 0.5 \times SR$
16:        if $f(x) \leq f(x^b)$ then
17:            $x_i^b \leftarrow x_i$
18:        end if
19:        if $f(x) > f(x^b)$ then
20:            $Restore(x_i)$
21:        else
22:            $Improve \leftarrow TRUE$
23:        end if
24:    else
25:        $Improve \leftarrow TRUE$
26:    end if
27: end for
```

(line 17) and *Improve* gets a *TRUE* value. Otherwise, the old x_i^b value is restored and continue to the next x_i^b (incrementing i) until all the values of the vector $\vec{x}^b$ were tested.

The LS1 Algorithm 1 used in our micro-DE, is the same as the original proposal [6]. As we can see, Algorithm 1 line 3, only the lower limit for the search range *SR* is modified with regard to the original algorithm. Were *MAX* correspond to the upper limit of the search range and *MIN* to the lower limit.

III. Micro Differential Evolution algorithm with Local Search (micro-DELS)

Our micro-DELS uses *Np* = 5, *mutation*, and *combination* mechanisms typical in DE as well as the F and *CR* parameters, but not all have the same job in DE. It is important to note the role of individuals in the population for our micro-DELS. We propose that, at the beginning of each restart cycle are split in two kinds:

1) Elitist solutions
2) The remaining

The elitist individuals are the two with the best aptitude from previous restart cycle and are used to preserve the best vectors to pass to next generation. The remaining individuals are manipulated in the Restart Population procedure, the third and fourth individual are used for local exploration and the fifth individual used for maintain diversity.

We have incorporated four mechanisms to the micro-DE algorithm with the aim of enhancing its efficiency:

1) First, the two individuals with the best aptitude pass to the next generation as the elitist individuals, the best and the second best individuals.
2) Second, we design two $tracking - vectors$ named $\vec{w}_1$ and $\vec{w}_2$, in order to follow the best individual by a decreasing random distance for local exploration.
3) Third, $restart - vector$ named $\vec{q}$ for maintaining a high diversity and global exploration.
4) Fourth, a local search mechanism based on the LS1 of MTLS for exploitation and for maintain diversity is applied on the best individual.

The $\vec{w}$ vectors are generated by using the best individual, $\vec{x}_i^b$, as reference. $\vec{w}$ are generated as follows:

$$\vec{w}_i = \vec{x}^b \times R_d \times \lambda \tag{7}$$

where $R_d \leftarrow R_d \times 0.9$ and is different for each $\vec{w}$. Also R_d is initialized at the beginning of the algorithm and changes when the $RestartPopulation$ procedure is called D times, with D as the problem dimension. We assume $\lambda \in [0, 1]$ as a real random number following a uniform distribution.

The vector:

$$\vec{q} = (q_1, q_2, ..., q_D) \tag{8}$$

is initialized with

$$\vec{q}_i \leftarrow rand[\mathit{MIN}, \mathit{MAX}] \tag{9}$$

where *MIN* and *MAX* are the valid limits of the search space.

As we said before, the local search mechanism that we use is the $LS1$ as it appears in [5], but the search range *SR* limit is changed from its original value:

$$\textit{From } SR < 10^{-15} \quad to \quad SR < 10^{-5} \tag{10}$$

This change in the *SR* limit improves the local search time by decreasing the search range in the lower bound.

LS1 is applied only to the best individual, to allow the exploitation of its local search space. If the solution is not improved, the previous value is restored.

Algorithm 2 shows the micro-DELS pseudo code. It starts by initiating R_d and *CR*. Line 3 $GenerateInitialPopulation$ procedure, who generates DE initial population randomly sampled as follows:

$$\vec{x}_i \leftarrow rand[\mathit{MIN}, \mathit{MAX}], \quad for \quad i = 1, ..., D \tag{11}$$

We assume a uniform distribution of the initial $\vec{x}_i$ values generated. The $EvaluateInitialPopulation$ function (Algorithm 2 line 4) evaluates all initial population individuals and obtains its aptitude for further algorithm comparisons.

The essential part of the micro-DELS algorithm (Algorithm 2, lines 7 to 14) initiates at the while loop (Algorithm 2, line 5) and consists of the following four functions: $CalculateTrialVector$, $SortPopulation$, $RestartPopulation$ and $LocalSearch$:

Algorithm 2 Pseudocode for micro-DELS

```
 1: $R_d \leftarrow Rini$
 2: CR$\leftarrow 0.001$
 3: $P_0 \leftarrow$ GenerateInitialPopulation()
 4: EvaluateInitialPopulation ($P_0$)
 5: while NOT TerminationConditionMeet do
 6:     $F \leftarrow random[0.1, 1.5]$
 7:     for $n = 1$ to Np do
 8:         CalculateTrialVector($\vec{x}_{i,G}$)
 9:     end for
10:     SortPopulation($P_G$)
11:     if $RestartCount = 5$ then
12:         RestartPopulation($\vec{q}_G, \vec{w}_{1,G}, \vec{w}_{2,G}$)
13:         if $Ls1Count = 50$ then
14:             LocalSearch1($\vec{x}^b_G$)
15:         end if
16:     end if
17: end while
```

- $CalculateTrialVector()$: The basis of a DE algorithm, it generates trial vectors and compares it against a pre-selected population individual, see eq. 3 and 5.
- $SortPopulation()$: From best to worst.
- $RestartPopulation()$: Restarts the random individual, and generates the tracking ones, as in eq. 7, 8 and 9.
- $LocalSearch1()$: Local search procedure, described in Algorithm 1.

Because its small population, micro-DELS can do little adjustments to one individual, search in local optima and by the use of the tracking individual make local exploration. The use of the LS1 for each component of the best individual exploits the bests founded solutions. Because LS1 consumes D FE's every time is used, we make empirical test to find the best way and period of calling it and found that every 50 of a restart population loop (line 12) gives the best result.

The best performance is obtained with two elitist individuals. In micro-DE diversity is achieved with the phases of *mutation* and *combination*, as well as the individuals with random values.

IV. Experiments

This section presents the experimental methodology used to evaluate and compare our micro-DELS performance.

Experiments are presented in two sections. A first part tests the LSGO of the CEC2012 benchmark with a canonical Differential Evolution performance with regards to the same tests to our micro-DELS. The detailed description of the functions used can be found in [2]. In this part we report results and show a comparative as well as convergence plots for both algorithms.

In the second part, we present the comparison of the results for the benchmark reference algorithms with regard to our micro-DELS proposal, and a statistical analysis to three algorithms. Also we show the results of the statistical analysis.

TABLE I. PARAMETER SETTINGS

Parameter	DE	micro-DELS
Max runs number	25	25
FE' s	3,000,000	3,000,000
Population size	**60**	**5**
F	0.1 to 1.5	0.1 to 1.5
CR	0.001	0.001

TABLE II. 20 BENCHMARK FUNCTIONS

Name	Separable	Rotated	Min.	Max.
F1 - Sh. Elliptic	Y	N	-100	100
F2 - Sh. Rastrigins	Y	N	-5	5
F3 - Sh. Ackleys	Y	N	-32	32
F4 - Sh. & m-rot. Elliptic	N	Y	-100	100
F5 - Sh. & m-rot. Rastrigins	N	Y	-5	5
F6 - Sh. & m-rot. Ackleys	N	Y	-32	32
F7 - Sh. m-dim. Schwefels P. 1.2	N	N	-100	100
F8 - Sh. m-dim. Rosenbrocks	N	N	-100	100
F9 - D/2m-gp. Sh. & m-rot. Elliptic	N	Y	-100	100
F10 - 2m-gp. Sh. & m-rot. Rastrigins	N	Y	-5	5
F11 - 2m-gp. Sh. & m-rot. Ackleys	N	Y	-32	32
F12 - 2m-gp. Sh. m-dim. Schwefels P. 1.2	N	N	-100	100
F13 - 2m-gp. Sh. m-dim. Rosenbrocks	N	N	-100	100
F14 - Sh. & m-rot. Elliptic	N	Y	-100	100
F15 - Sh. & m-rot. Rastrigins	N	Y	-5	5
F16 - Sh. & m-rot. Ackleys	N	Y	-32	32
F17 - Sh. m-dim. Schwefels P. 1.2	N	N	-100	100
F18 - Sh. m-dim. Rosenbrocks	N	N	-100	100
F19 - Sh. Schwefels P. 1.2	Y	N	-100	100
F20 - Sh. Rosenbrocks	Y	N	-100	100

Our proposal is a micro-DE algorithm adjusted (*CR = 0.001*) for large scale optimization that uses DE mechanisms. Its behavior is different, from all previous micro-DE algorithms, and achieves better improvements, per FE's, than canonical DE (i.e., for F20 minimization DE achieves 5.76e+05 and micro-DELS = 9.87e+02) in the same functions at the LSGO testbed of the CEC2012.

A. Experimental Configuration

The settings for the first step of experiments, used for the two algorithms, is the same, see table I, except for the population size. The algorithms DE and micro-DELS use the variant DE/rand/1/bin.

The experimental setting of our micro-DELS, for the second part of the study is: $CR \leftarrow 0.001$, $F \in [0.1, 1.5]$ (for F values adjust see [15]) and population size of five. The canonical DE and the micro-DE were programmed in C code and compiled by using g++. As well the CEC2012 C++ libraries given by Tang et al. see [13].

B. Benchmark Used

The benchmark used in this article is a set of 20 Functions from the CEC2012 Special Session and Competition on Large-Scale Global Optimization, LSGO, see [13]. In table II we show the set of functions used and their main characteristics: functions name, separability, rotated, as well as the upper and lower limits of variables. All functions there will be minimized and the optimal value is zero.

V. RESULTS

This section reports, first the performance of canonical DE and our micro-DELS, second, the benchmark reference algorithms compared to our proposals performance.

A. Canonical DE and micro-DELS

Here we present a comparison between canonical DE and micro-DELS. We can see the results (convergence plots) in figs 1-3. Three characteristic curves will be showed F5, F8 and F10. We show the average results over 25 runs in all plots.

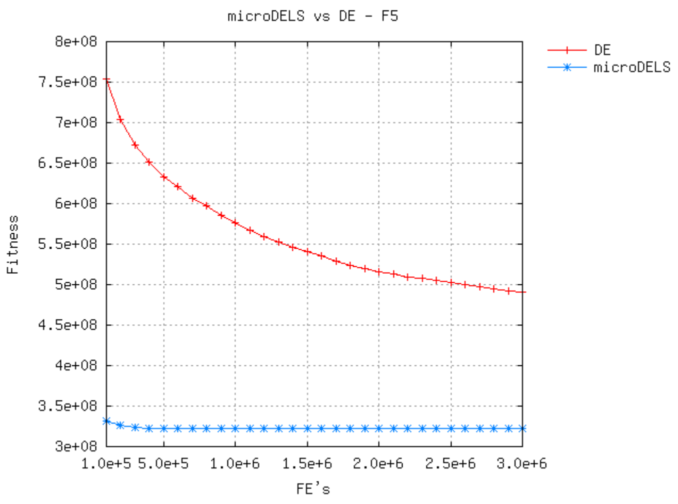

Fig. 1. Convergence of the mean value for function F5.

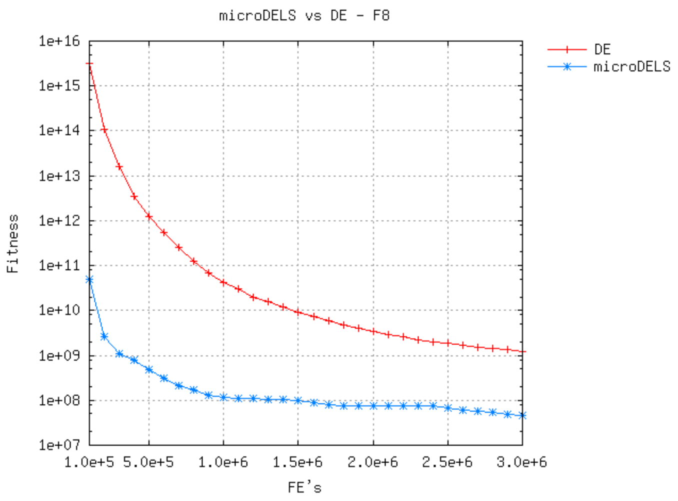

Fig. 2. Convergence of the mean value for function F8. Logarithmic Y scale

Some characteristics can be highlighted from the three convergence plots:

1) micro-DELS starts from a better value.
2) micro-DELS is always under DE in the plots.
3) micro-DELS always gets a better final result, from one (fig. 2) to 3 (figs. 1 and 3) orders of magnitude better.

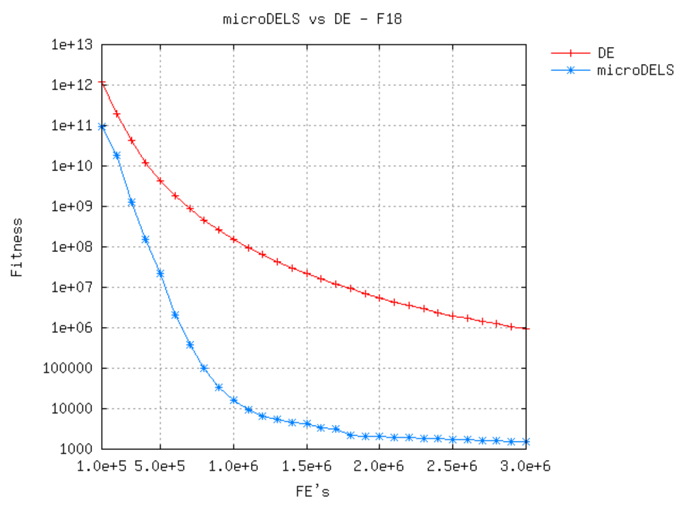

Fig. 3. Convergence of the mean value for function F18. Logarithmic Y scale

TABLE III. MICRO-DELS VS. CEC2012 REFERENCE

Function	DE	DECC-CG	MLCC	micro-DELS
F1	1.28e+07	2,86e-07	**0,00e+00**	1,16E+00
F2	6.22e+02	1,31e+03	**6,43e-11**	3,74E-03
F3	2.70e+00	1,39e+00	**1,46e-13**	1,78E-03
F4	9.03e+13	1,51e+13	1,03e+13	**4,89E+12**
F5	4.94e+08	**2,38e+08**	3,92e+08	3,28E+08
F6	2.04e+07	**4,80e+06**	1,95e+07	1,93e+07
F7	6.43e+10	1,07e+08	**5,15e+05**	6,86E+08
F8	1.24e+09	6,70e+07	4,67e+07	**2,30E+07**
F9	2.03e+09	3,18e+08	**1,24e+08**	1,86E+08
F10	7.58e+03	1,07e+04	**3,16e+03**	3,76E+03
F11	2.13e+02	**2,33e+01**	1,98e+02	1,94e+02
F12	1.29e+06	8,87e+04	3,47e+04	**3,40E+04**
F13	9.38e+04	3,00e+03	**1,91e+03**	2,58E+03
F14	5.32e+09	8,07e+08	**3,16e+08**	1,06E+09
F15	1.59e+04	1,18e+04	**6,89e+03**	7,49E+03
F16	4.20e+02	**7,51e+01**	3,95e+02	3,88+02
F17	3.20E+06	2,89e+05	1,59e+05	**1,50E+05**
F18	9.53e+05	2,30e+04	4,17e+03	**1,57E+03**
F19	1.29e+07	**1,11e+06**	1,36e+06	2,30E+06
F20	5.76e+05	3,98e+03	2,04e+03	**9,87E+02**

In Table III we show the average distribution errors obtained for canonical DE and our micro-DELS. It is clear that micro-DELS outperforms a canonical DE in the whole testbed.

As we can see, micro-DELS reaches twenty out of twenty better values than the DE reference algorithm. In some hard to optimize functions, micro-DELS is better than DE only for a few units in the same order of magnitude like functions: F5, F6, F10, F11, and F16. However, the average performance of our micro-DELS achieves one, two or three orders of magnitude better accuracy than DE as in: F3, F4, F7, F8, F9, F12, F13, F14, F15, F17, F18, F19 and F20.

We also highlight that in two functions our micro-DELS achieves five (F2) or even six (F1) orders of magnitude better accuracy than DE.

B. DECC-CG, MLCC, and micro-DELS

Table III contains the average of distribution errors obtained for DECC-CG, MLCC, and our micro-DELS, after 25 independent runs of the 20 functions test bed from LSGO special session of CEC12.

We can see that our micro-DELS obtains the second best value in ten of the twenty functions, and remains in third place in four functions. Also, our micro-DELS achieves six better values than the reference algorithms: F4, F8, F12, F17, F18 and F20. (see table III).

VI. ANALYSIS OF RESULTS

This section is divided into two parts. In the first part we make the analysis of the three algorithms performance according to the rules used in the LSGO competition. In the second part we make a statistical comparison of the algorithms performance.

A. Numerical Results

In this part we use the rules of the LSGO CEC2012, i.e., the algorithms of the experiments where organized according to their decreasing mean value. We use ranks from 1(best) to 3(worst), so that place 1 is rewarded with 25 points, place 2 with 18, and place 3 with 15.

TABLE IV. COMPARISONS OF ALGORITHMS

Algorithm	I	II	III	IV	V	Sum.
MLCC	**75**	91	**108**	**101**	36	411
micro-DELS	51	**101**	97	**101**	**40**	390
DECC-CG	48	98	85	88	**40**	359

The results of comparison is presented in table IV, where our micro-DELS gets the better results in column II of the comparison (from functions F4 to F8) where all the functions are non separable and three are multiplied by a rotated matrix. In columns IV and V our micro-DELS is tied for first place, in columns IV with MLCC and in column V with DECC-CG. Our micro-DELS gets three best results out of five sections of the benchmark. Also, we must highlight the fact that micro-DELS does not ends in the last place in any section of the benchmark, so for this benchmark it ends as the more reliable.

B. Statistical Analysis

In this section we make a non-parametrical statistical test. These tests are used to find out if the numerical distributions of the results follow the conditions of normality and homocedasticity, [10]. In this paper we use the Friedman ranking test as well as Holm multicompare as post-hoc to statistically determine their relation to the reference one.

As we can see in Table V, the Friedman test [7], is used as an *a priori* test to rank the algorithms results. The Friedman test assigns rankings to the obtained results for each j algorithm on each i problem, so we can see from the best algorithm position (1) to the last.

TABLE V. AVERAGE RANKINGS OF THE ALGORITHMS (FRIEDMAN)

Algorithm	Ranking
DECC-CG	2.42
MLCC	1.72
micro-DELS	**1.85**

C. Post hoc Comparison (Friedman)

In Table VI we present a post hoc comparison over the results of Friedman test showing the p-values obtained.

TABLE VI. POST HOC COMPARISON TABLE FOR $\alpha = 0.05$ (FRIEDMAN)

i	algorithm	$z = (R_0 - R_i)/SE$	p	Holm Hochberg
2	DECC-CG	2.213594	0.026857	0.025
1	micro-DELS	0.395285	0.692633	0.05

Holm's procedure rejects those hypotheses that have a p-value ≤ 0.025. Therefore, DECC-CG has a big difference with regard to the control algorithm (MLCC). On the other hand, micro-DELS has the minor difference with respect to MLCC.

D. Adjusted p-values (Friedman)

By using the adjusted p-values obtained through the application of the post hoc method, it is clear that our micro-DELS is statistically equivalent to MLCC, while DECC-CG is a worse algorithm.

TABLE VII. ADJUSTED p-VALUES (FRIEDMAN)

i	algorithm	unadjusted p	p_{Holm}	$p_{Hochberg}$
1	DECC-CG	0.026857	0.053713	0.053713
2	micro-EDLS	0.692633	0.692633	0.692633

VII. CONCLUSIONS

In this article our proposal is to incorporate a local search procedure into a micro-DE algorithm with the aim of improving the accuracy of this algorithm for high dimensional problems. Results clearly show that micro-DELS is more competitive for high dimensional problems than canonical DE.

Our proposal was empirically tested in a set of twenty high dimensional complex functions, then our micro-DELS was compared against DECC-CG and MLCC, the reference algorithms of this benchmark. The micro-DELS obtained six best results out of twenty. In the rest it was equivalent to them. From the results of this experiments we can extract the next conclusions:

After the shown results we can claim that our micro-DELS is statistically better than canonical DE . It is also either better or equivalent to the state of the art DECC-CG and MLCC, respectively.

As a future work we are interested in adjusting the *CR* parameter of our micro-DELS as well as the $tracking$ and $reset$ individuals, to obtain a better performance. Also, we plan to use our micro-DELS for real problem applications where its light cost could be a key advantage over other techniques. We plan to actually go to the performance bounds of the technique, including a scalability analysis for our next work.

ACKNOWLEDGMENTS

Authors would like to thank:

- "Instituto Politécnico Nacional" specially, but not limited, for its work permission with register number SeAca/COTEPABE/79/12.
- "Consejo Nacional de Ciencia y Tecnología, CONACyT". For its economical support for the project with register number 175589.
- Authors acknowledge funds from the Spanish Ministry of Economy and Competitiveness (MEC) and FEDER under contract TIN2011-28194 (roadME project, http://roadme.lcc.uma.es).
- "Instituto Politécnico Nacional" in association with "Consejo Nacional de Ciencia y Tecnología, CONACyT". For its economical support in the program "Becas Mixtas 2012-2013 Movilidad en el extranjero" with register number 290674

REFERENCES

[1] E.G. Johnson and M.A.G. Abushagur: *Micro-Genetic Algorithm Optimization Methods Applied to Dielectric Gratings*. Journal of the Optical Society of America, 12(5):1152-1160, 1995.

[2] F. Herrera, M. Lozano, and D. Molina: *Test suite for the special issue of soft computing on scalability of evolutionary algorithms and other metaheuristics for large scale continuous optimization problems*. Technical report, SCI2S, University of Granada, Spain, May 2010.

[3] J. Manuel Garcia-Nieto: *Emergent Optimization: Design and Applications in Telecommunications and Bioinformatics*, Phd Doctoral Thesis. University of Malaga, Spain February 2013.

[4] Krishnakumar, K.: *Micro-Genetic Algorithms for Stationary and Non-Stationary Function Optimization*, In SPIE Proceedings: Intelligent Control and Adaptive Systems, Philadelphia, PA, Vol 1196, pp 289-296. Year 1989.

[5] K. Tang, X. Yao, P. N. Suganthan, C. MacNish, Y. P. Chen, C. M. Chen, and Z. Yang: *Benchmark functions for the CEC08 special session and competition on large scale global optimization*. Technical report, Nature Inspired Computation and Applications Laboratory, USTC, China, November November 2007.

[6] L. Tseng and C. Chen: *Multiple trajectory search for unconstrained/-constrained multi-objective optimization*. In Proceedings of the Eleventh conference on Congress on Evolutionary Computation, CEC09, pages 19511958, Piscataway, NJ, USA, IEEE Press, 2009.

[7] M. Friedman: *A comparison of alternative tests of significance for the problem of m rankings*, Annals of Mathematical Statistics, pp. 8692, 1940.

[8] Parsopoulos, Konstantinos E.: *Cooperative MicroDifferential Evolution for HighDimensional Problems*, In Proceedings of the GECCO09, Montral Qubec, Canada. ACM 978-1-60558-325-9/09/07, pp 531-538. 2009.

[9] R. Storn, K. Price: *Differential Evolution - a Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces*. Technical Report TR-95-012, ICSI, March 1995.

[10] S. Garc, D. Molina, and F. Herrera: *A tutorial on using nonparametric statistical test for multiple comparisons of metaheuristics and evolutionary algorithms*, in MAEB, pp. 455462, 2012.

[11] Swagatam Das and Ponnuthurai Nagaratnam Suganthan: *Differential Evolution: A Survey of the State-of-the-Art*, in *IEEE Transactions on Evolutionary Computation*, Vol. 15, No. 1, February 2011.

[12] T. Huang and A. S. Mohan: *Microparticle swarm optimizer for solving high dimensional optimization problems (PSO for high dimensional optimization problems*. Applied Mathematics and Computation, 181(2):11481154, 2006.

[13] Tang, K., Li, X., Suganthan, P.N., Yang, Z., Weise, T.: *Benchmark functions for the CEC2010 special session and competition on large scale global optimization*. TR, NICAL, USTC, Hefei, Anhui, http://nical.ustc.edu.cn/cec10ss.php, China, 2009.

[14] Yunyoung Kim*, Koji Gotoh*, Masahiro Toyosada*, and Jewoong Park: *Micro-Genetic Algorithms(GAs) for Hard Combinatorial Optimisation Problems*, In Proceedings of The Twelfth (2002) International Offshore and Polar Engineering Conference, ISBN 1-880653-58-3 (Set); ISSN 1098-6189 (Set) , 2002.

[15] Zaharie, D. (2002). "Critical values for the control parameters of differential evolution algorithms". Proceedings of the 8th International Conference on Soft Computing (MENDEL). Brno, Czech Republic. pp. 62-67