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

Large Scale Global Optimization: Experimental Results with MOS-based Hybrid Algorithms

Antonio LaTorre*†, Santiago Muelas*, José-María Peña*
* DATSI, Facultad de Informática, Universidad Politécnica de Madrid
† Instituto Cajal, Centro Superior de Investigaciones Científicas (CSIC)

***Abstract*—Continuous optimization is one of the most active research lines in evolutionary and metaheuristic algorithms. Through CEC 2005 to CEC 2013 competitions, many different algorithms have been proposed to solve continuous problems. The advances on this type of problems are of capital importance as many real-world problems from very different domains (biology, engineering, data mining, etc.) can be formulated as the optimization of a continuous function. In this paper we describe the whole process of creating a competitive hybrid algorithm, from the experimental design to the final statistical validation of the results. We prove that a good experimental design is able to find a combination of algorithms that outperforms any of its composing algorithms by automatically selecting the most appropriate heuristic for each function and search phase. We also show that the proposed algorithm obtains statistically better results than the reference algorithm DECC-G.**

***Keywords*—Continuous Optimization, Large Scale Global Optimization, Hybridization, MOS, MTS, MTS-LS1-Reduced, Solis and Wets, GA, DE, GODE**

I. Introduction

Continuous optimization is getting more and more attention in the last years. Many real-world problems from different domains (biology, data mining, engineering, etc.) can be formulated as the optimization of a continuous function. These problems have been tackled using Evolutionary Algorithms (EAs) [1] or similar metaheuristics [2]. Selecting an appropriate algorithm to solve a continuous optimization problem is not a trivial task. Although a particular algorithm can be configured to perform properly in a given scale of problems (considering the number of variables as their dimensionality), the behavior of the algorithm can degrade as this dimensionality increases, even if the nature of the problem remains the same [3].

In this contribution, we have studied the behavior of several well-known optimization algorithms, both population-based and local searches, as well as two algorithms proposed for this work, on the benchmark of problems proposed for the *"Special Session on Large Scale Global Optimization"* held at the IEEE CEC 2013 [4]. Each algorithm has been subject to a parameter tuning to find the most suitable combination of parameters for the proposed problems. Then, the best performing algorithms among these have been selected for their combination within the Multiple Offspring Sampling (MOS) framework. This framework allows the combination of different metaheuristics following a HRH (High-level Relay Hybrid) approach (this nomenclature will be reviewed in Section II) in which the number of evaluations that each algorithm can carry out is dynamically adjusted. Analogously to the case of the individual algorithms, the hybrid approach has also been subject to a parameter tuning, to find the best performing hybridization strategy for this benchmark and algorithms. Finally, the selected hybrid combination has been statistically compared with each of its composing algorithms as well as with the reference algorithm proposed by the organizers of the Special Session.

The remainder of this paper is organized as follows. Section II briefly reviews some relevant work conducted on similar problems. In Section III the individual algorithms used in the experimentation as well as the MOS framework are briefly introduced, whereas section IV presents the experimental scenario under study, describing both the considered benchmark and the execution environment used in our tests. In Section V, the results obtained are presented and discussed, listing the main conclusions derived from this study. Finally, Section VI contains the concluding remarks obtained from this work.

II. Preliminaries

The HRH terminology was introduced in [5], one of the first attempts to define a complete taxonomy of hybrid metaheuristics. This taxonomy is a combination of a hierarchical and a flat classification structured into two levels. The first level defines a hierarchical classification in order to reduce the total number of classes, whereas the second level proposes a flat classification, in which the classes that define an algorithm may be chosen in an arbitrary order. From this taxonomy, the following four basic hybridization strategies can be derived: (a) LRH (Low-level relay hybrid): One metaheuristic is embedded into a single-solution metaheuristic. (b) HRH (High-level relay hybrid): Two metaheuristics are executed in sequence. (c) LTH (Low-level teamwork hybrid): One metaheuristic is embedded into a population-based metaheuristic. (d) HTH (High-level teamwork hybrid): Two metaheuristics are executed in parallel. For this work, we have focused on the HRH group, the one the proposed algorithm belongs to.

In the last years there has been an intense research in HRH and, in particular, in memetic models, combining different types of metaheuristics. A good example of this is that, in most of the sessions that have focused on large scale global optimization recently (CEC 2008, ISDA 2009, CEC 2010, CEC 2012; and also a special issue in the "Soft Computing - A Fusion of Foundations, Methodologies and Applications" journal) most of the best performing algorithms were memetic algorithms [2], [3], [6]–[8]. For this reason, a HRH algorithm combining algorithms of different nature has been studied in this paper.

Finally, for an updated survey on memetic algorithms in the field of large scale global optimization, the readers are referred to [9], [10].

III. Proposed Approach

This section describes, first, the individual algorithms that have been considered in our experimentation and, then, the hybridization approach used in our study.

A. Individual Algorithms

1) Genetic Algorithm: A Genetic Algorithm (GA) has been used in our experiments. In particular, we have used the $BLX-\alpha$ crossover, with $alpha = 0.5$, and the Gaussian mutation [11]. Different population sizes and operator probabilities were tested, as described in Section IV.

2) Differential Evolution: A Differential Evolution (DE) algorithm [12] with Exponential Crossover was used in the experimentation. In the aforementioned special sessions and competitions, there has always been a DE (or variant) among the best performing algorithms. As for the GA, different population sizes and F and CR constants were used.

3) Self-Adaptive Differential Evolution: A Self-Adaptive Differential Evolution algorithm [13] has also been used. In this case, the same parameters as for the DE have been considered, as well as two new parameters, τ_F and τ_{CR}, which determine the probability to adjust the F and CR parameters, respectively.

4) Generalized Opposition-Based Differential Evolution: The fourth algorithm used in our experiments was the Generalized Opposition-Based Differential Evolution (GODE) algorithm [14]. This algorithm uses the Opposition-Based Learning concept which basically means that, with some probability, the algorithm tries not only to evaluate the solutions in the current population, but also the opposite ones. Apart from the common DE parameters, this algorithm has been tuned for the probability of applying the opposite search.

5) Self-Adaptive Generalized Opposition-Based Differential Evolution: This algorithm, proposed specifically for this experimentation, combines the principles of the two previously presented algorithms to try to exploit the benefits of the self-adaptation of the DE control parameters and the opposite search. Consequently, the parameter tuning affects the parameters of both algorithms.

6) Solis and Wets Algorithm: The well-known Solis and Wets algorithm [15] has also been included in our study. This direct search method performs a randomized local minimization of a candidate solution with an adaptive step size. It has several parameters that rule the way the candidate solution is moved and how the step size is adapted. All these parameters will be subject to parameter tuning, as for the rest of the algorithms.

7) MTS-LS1 Algorithm: MTS-LS1 is the first of the three local searches combined by the MTS algorithm [2]. It has been successfully used in the past to search large scale global optimization problems. In the same line than the Solis and Wets algorithm, several parameters control the movements of the local search, and all of them will be adjusted.

8) MTS-LS1-Reduced Algorithm: MTS-LS1-Reduced is a new local search algorithm specifically proposed for this study. It is a modification to the MTS-LS1 algorithm that tries to optimize the number of evaluations consumed. The MTS-LS1-Reduced algorithm conducts the same operations as the MTS-LS1 local search but, instead of exploring all the dimensions, it spends most of its efforts in the most promising dimensions. At each step, the MTS-LS1-Reduced algorithm stores, for each dimension, the score improvements obtained when exploring that dimension, i.e., $max(0, score_{old} - score_{new})$ (for minimization problems). This information is used at the beginning of the next step to determine the dimensions to explore. MTS-LS1-Reduced uses two parameters for conducting this task: *improvePerc*, a percentage value used to select the dimensions that represent the *improvePerc* percent of the stored score improvements and *minPerc*, the percentage of other dimensions (selected randomly) that are going to be explored. By means of the *minPerc* parameter, MTS-LS1-Reduced can explore other dimensions to detect potentially better dimensions for the next steps and avoids falling into an excessive exploitative behavior.

B. Multiple Offspring Sampling

Multiple Offspring Sampling (MOS) is a framework for the development of Dynamic Hybrid Evolutionary Algorithms [16]. It has been successfully applied to combine different types of algorithms on well-known continuous optimization benchmarks [3], [17].

The MOS framework allows the seamless combination of several algorithms (both population-based and local searches) in a dynamic way. This means that the participation of each algorithm (the number of new candidate solutions that each algorithm is allowed to create) is adjusted dynamically according to some quality measure, such as, for example, the average fitness increment of the newly created individuals, that was the measure used in this proposal. Moreover, the proposed hybrid algorithm has followed a HRH approach, which means that algorithms are used in sequence, one after the other, each of them reusing the output population of the previous algorithm. This approach fits better when there are non-population-based techniques, such as local searches, as techniques are not constrained to produce a % of the common population, which is the case of the HTH approach. The overall search process is thus divided into different steps (blocks of a fixed number of Fitness Evaluations (FEs)) and the participation of each algorithm for step $i+1$ is adjusted according to its performance on the previous step i. Due to the limited space in this paper, we refer the reader to a more detailed description of the MOS framework that can be found in [16].

IV. Experimental Setup and Parameter Tuning

For the experimentation, the benchmark from the *"Special Session on Large Scale Global Optimization"* held at the IEEE CEC 2013 has been considered. This benchmark defines 15 continuous optimization functions with different degrees of separability: from completely separable functions to fully non-separable functions. Detailed information on the benchmark can be found in [4].

In order to find the most suitable hybrid algorithm for the proposed benchmark, we have conducted a large experimentation to, first, find the most suitable individual algorithms among those described in Section III and, second, find the best strategy to combine them.

Table I contains the parameters and the tested values for all the individual techniques. In particular, we have tested several values for the following parameters, which can dramatically influence on the performance of the algorithm. For the DE algorithm, we have explored several population sizes (*popSize*), *F* and *CR* constants. The same parameters have been considered for the Self-Adaptive DE algorithm, the GODE algorithm and the Self-Adaptive GODE algorithm. However, each of them has also some specific parameters that were subject to adjustment. For the GODE algorithm, the probability to apply the Opposition-Based search (*godeProb*) has been tuned. Regarding the Self-Adaptive DE, we considered the probabilities to conduct the adjustment on the F and CR constants (τ_F and τ_{CR}). Finally, for the Self-Adaptive GODE algorithm, both parameters from Self-Adaptive DE and GODE algorithms have been taken into account. Regarding the GA, we have adjusted the population size (*popSize*) and the probabilities for crossover and mutation (*pcx* and *pm*, respectively). For the Solis and Wets algorithm, we have focused on the maximum number of successful modifications of the solution (*maxSuccess*) before increasing the adjustment ratio (*delta*), the maximum number of unsuccessful modifications (*maxFailed*) and the increasing and decreasing factors (*adjustSuccess* and *adjustFailed*, respectively). On the other hand, regarding the MTS-LS1 algorithm, we have considered the resetting factors for the SR parameter after one iteration without improvements (*adjustFailed*) or after reaching the minimum value for SR (*adjustMin*) and the adjustment ratios in both directions of the search space (*moveLeft* and *moveRight*, respectively). Finally, the MTS-LS1-Reduced algorithm shares the same parameters of MTS-LS1 plus the improvements percentage to focus the search (*improvePerc*) and the minimum percentage for the remaining dimensions to be explored (*minPerc*). The values tested in this experimentation have been selected according to previous studies [6], [8], [18] and a preliminary experimentation on this benchmark.

For the experimentation, a fractional design based on orthogonal matrices according to the Taguchi method [19] has been carried out. In this method, the concept of signal to noise ratio (SN ratio) is introduced for measuring the sensitivity of the quality characteristic being investigated in a controlled manner to those external influencing factors (noise factors) not under control. The aim of the experiment is to determine the *highest* possible SN ratio for the results since a high value of the SN ratio implies that the signal is much higher than the random effects of the noise factors. From the quality point of view, there are three possible categories of quality characteristics: (i) smaller is better, (ii) nominal is best and (iii) bigger is better. The obtained results fall in the "smaller is better category" since the objective is to reduce the error between the best solution found and the global optimum. For this category, the SN ratio estimate is defined in Eq. 1, where n denotes the total number of instances and $y_1, y_2, \ldots, y_n$ the target values (the error to the best solution in this case).

$$SN = -10 \log \left(\frac{1}{n} \sum_{t=1}^{n} y_t^2 \right) \quad (1)$$

This method allows the execution of a limited number of configurations and still reports significant information on the best combination of parameter values. In particular, a maximum of 27 different configurations were tested for each algorithm on the whole set of functions (the exact number will depend on the number of parameters to be tuned). The values for the parameters of the best configuration appear in bold in Table I.

TABLE I. PARAMETERS VALUES FOR INDIVIDUAL ALGORITHMS

Parameter Values of DE	
popSize	**25**, 50, 100
F	**0.1**, 0.5, 0.9
CR	0.1, 0.5, **0.9**
Parameter Values of Self-Adaptive DE	
popSize	**25**, 50, 100
τ_F	0.05, 0.1, **0.2**
τ_{CR}	0.05, 0.1, **0.2**
Parameter Values of GODE	
popSize	**25**, 50, 100
F	**0.1**, 0.5, 0.9
CR	0.1, 0.5, **0.9**
godeProb	0.2, **0.4**, 0.8
Parameter Values of Self-Adaptive GODE	
popSize	**25**, 50, 100
F	**0.1**, 0.5, 0.9
CR	0.1, 0.5, **0.9**
τ_F	0.05, 0.1, **0.2**
τ_{CR}	0.05, 0.1, **0.2**
godeProb	0.2, **0.4**, 0.8
Parameter Values of GA	
popSize	100, 200, **400**
pcx	**0.01**, 0.05, 0.1
pm	0.1, 0.5, **0.9**
Parameter Values of Solis and Wets	
maxSuccess	2, **5**, 10
maxFailed	1, 3, **5**
adjustSuccess	2, **4**, 6
adjustFailed	0.25, 0.5, **0.75**
delta	0.6, 1.2, **2.4**
Parameter Values of MTS-LS1	
(initial) SR	50% of the search space
(min) SR	$1e-14$
adjustFailed	2, 3, **5**
adjustMin	**2.5**, 5, 10
moveLeft	**0.25**, 0.5, 1
moveRight	0.25, **0.5**, 1
Parameter Values of MTS-LS1-Reduced	
(initial) SR	50% of the search space
(min) SR	$1e-14$
adjustFailed	**2**, 3, 5
adjustMin	2.5, 5, **10**
moveLeft	**0.25**, 0.5, 1
moveRight	0.25, **0.5**, 1
improvePerc	0.7, 0.8, **0.9**
minPerc	**0.025**, 0.05, 0.1

Figure 1 represents an example on how the Taguchi method works. In particular, it shows the main effects plot for the SN ratio results obtained with the GA. A main effect plot is a plot of the mean response values (the SN ratio for these graphs) at each level of a design parameter. This plot can be used to compare the strength of the effects of the values of the parameter. From these results it can be observed that the most important parameter for the GA is the crossover probability. Furthermore, there are also clear differences on the

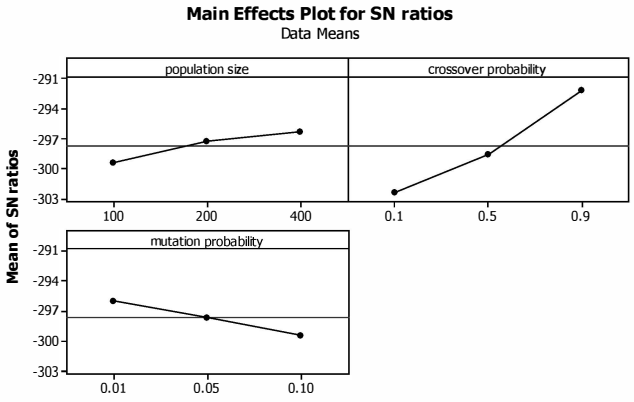

Fig. 1. Main Effects Plot for the SN ratios for the GA. As it can be observed, the parameter with more influence on the performance of the GA is the crossover probability (higher value is better). Furthermore, the other two parameters also show a clear performance bias to large population sizes and low mutation probability, respectively.

performance of the algorithm regarding the population size and the mutation probability. This plot reveals which combination of values for the parameters of the GA should be selected for further experimentation, as shown in Table I.

In order to compare the best configurations of all the individual algorithms across all the functions, the average rank according to the Friedman test [20] was computed for each algorithm. The nWins procedure [21] was also applied to the average error per function to perform a global comparative analysis. This procedure carries out a pair-wise statistical comparison over the distribution values of all the available algorithms by means of the Wilcoxon signed-rank test [22] with a confidence level of 0.05. With these results, the following analysis is carried out: if one algorithm is significantly better than other ($pvalue < 0.05$), the winning algorithm is granted +1 wins and the losing algorithm is penalized with -1 wins. The sum of all the "wins" constitutes the nWins value.

TABLE II. AVERAGE RANKING

	Ranking	nWins
MTS-LS1-Reduced	2.63	7
MTS-LS1	3.70	3
Solis and Wets	4.13	2
GA	4.47	1
Self-Adaptive DE	4.20	0
Self-Adaptive GODE	4.47	-1
GODE	6.13	-6
DE	6.27	-6

Table II presents the results of both rankings for each of the eight considered algorithms. On this table, we have marked in bold the algorithms that were selected for hybridization. According to these data, it is clear that the best performing individual algorithm is the MTS-LS1-Reduced, both in rank and nWins. The second best algorithm is the plain MTS-LS1 but, as it is a simplified version of the previous algorithm, it was discarded for its combination. The third algorithm is the Solis and Wets algorithm, whereas for the fourth place there are differences between the rank (Self-Adaptive DE) and the number of wins (GA). As the difference in the rank is very low and the GA has one more win than the Self-Adaptive DE, we preferred to select the GA for its combination. Regarding the remaining algorithms, as all of them are population-based algorithms and the required population sizes by each of them are too different to that of the GA, we decided to just consider these three algorithms for their hybridization: MTS-LS1-Reduced, Solis and Wets and GA.

TABLE III. PARAMETERS VALUES FOR THE HYBRID ALGORITHM

minPart	0%, 1%, 5%, 10%, **20%**
stepFactor	3000, 9000, 18000, 27000, **36000**

Analogously, Table III contains the parameters and the tested values for the hybrid algorithm combining the best configurations of the GA, the MTS-LS1-Reduced and the Solis and Wets algorithms. Concretely, we have studied two parameters: the minimum participation ratio for a technique (*minPart*) and the step factor (*stepFactor*). We have conducted the same fractional design as with individual techniques, and the final selected parameters are shown in bold.

In Section V we present the results of the MOS-based hybrid algorithm combining the three selected algorithms with the configuration reported in Tables I and III. In order to make the results comparable with other algorithms, we have strictly followed the conditions imposed by the benchmark. Therefore, for each combination, 25 independent executions were carried out. The stopping criterion, as defined in the benchmark, was a fixed number of fitness evaluations (3M FEs). The performance criterion (i.e. the response variable) is the distance (error) between the best individual found and the global optimum in terms of fitness value.

V. RESULTS AND DISCUSSION

In this section we present and discuss the results of the proposed algorithm on the benchmark used for this special session and competition. This analysis is divided into four parts. First, we show the convergence graphs for several functions in Section V-A. Second, we analyze the behavior of the MOS-based algorithm on the different groups of functions in Section V-B. Third, we statistically compare the proposed hybrid approach with its composing algorithms in Section V-C. Finally, Section V-D presents a statistical comparison of our results with those of the reference algorithm.

A. Convergence Analysis

In this section we provide the convergence graphs of the MOS-based algorithm on the six selected functions by the organizers of the special session: F_2, F_7, F_{11}, F_{12}, F_{13} and F_{14}. For each function, a single convergence curve has been plotted using the average results of 25 independent executions. Figures 2-7 show the convergence graphs for functions F_2-F_{14}, respectively. The following characteristics can be observed:

- In most of the functions (especially for F_2, F_7, F_{12} and F_{13}) we can see several search phases in which the slope of the convergence curve changes. This corresponds with an exchange on the current prevalent algorithm, as it can be clearly seen in Figures 8 and 9, that represent the participation plot for F_{12} and F_{13}, respectively, where each of the combined algorithms dominates at a different

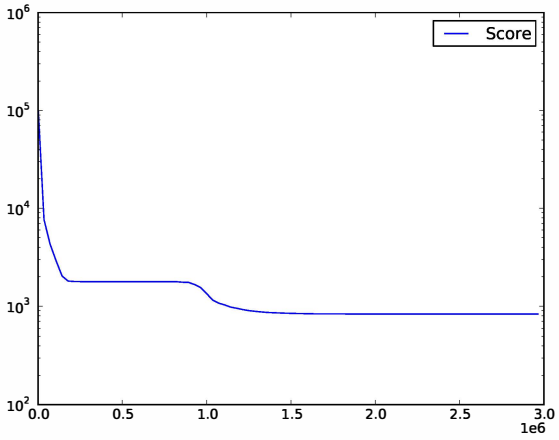

Fig. 2. Average convergence graph of 25 runs of function F_2

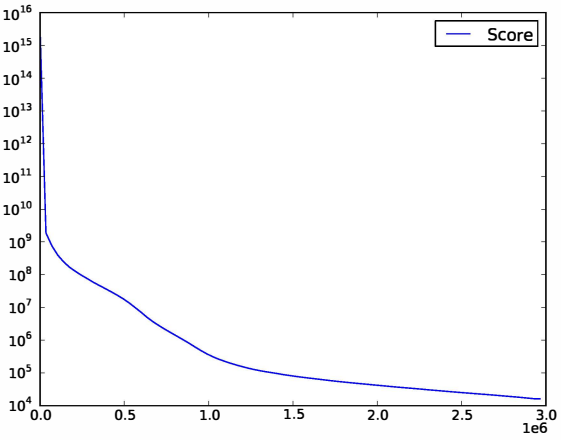

Fig. 3. Average convergence graph of 25 runs of function F_7

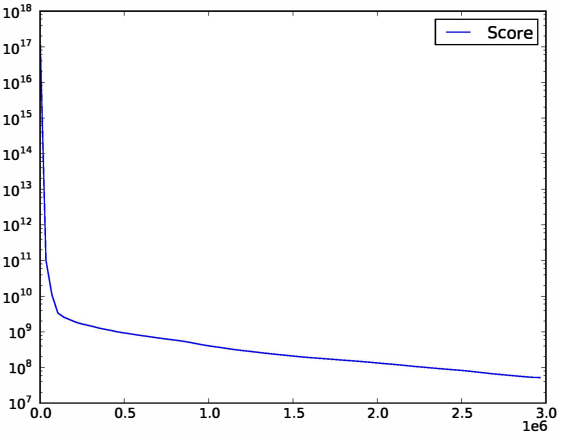

Fig. 4. Average convergence graph of 25 runs of function F_{11}

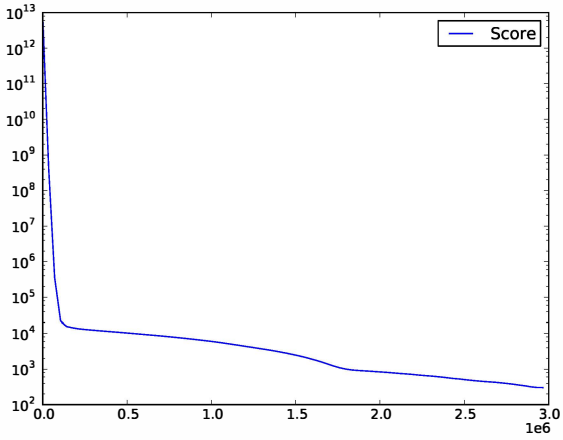

Fig. 5. Average convergence graph of 25 runs of function F_{12}

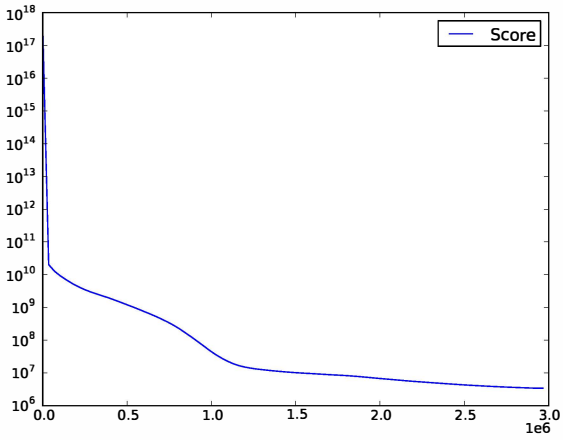

Fig. 6. Average convergence graph of 25 runs of function F_{13}

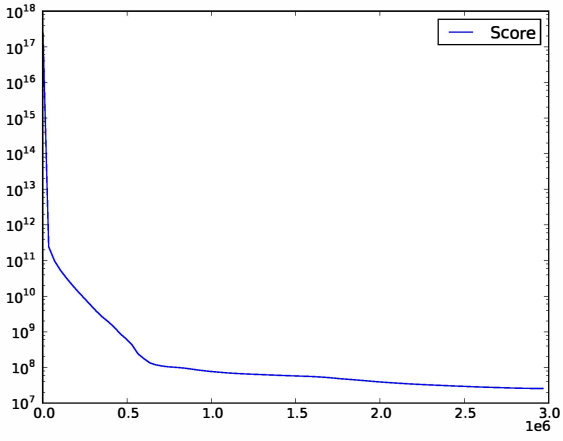

Fig. 7. Average convergence graph of 25 runs of function F_{14}

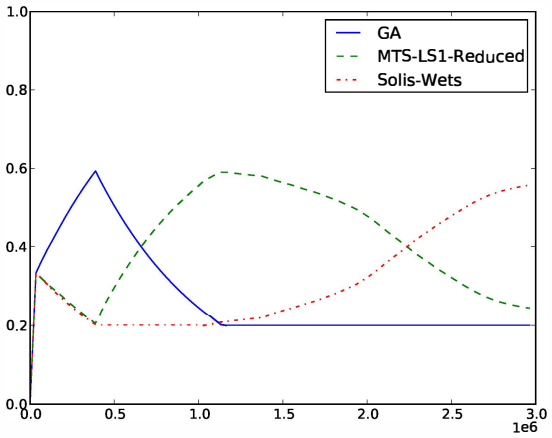

Fig. 8. Average participation plot of 25 runs of function F_{12}

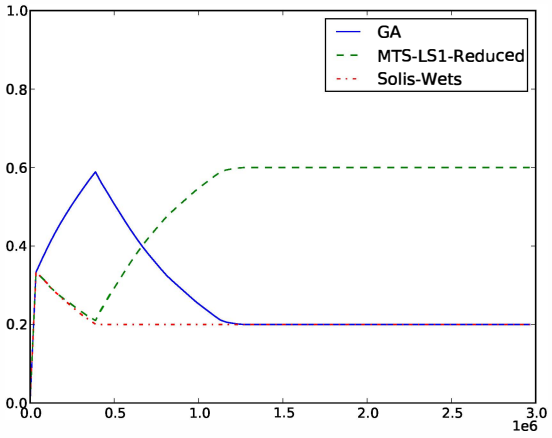

Fig. 9. Average participation plot of 25 runs of function F_{13}

stage of the search process. Similar plots can be obtained for the remaining functions.

- The hybrid algorithm does not seem to completely converge in none of the analyzed functions, as it continues to improve the best solution until the end of its execution.

B. Results of the MOS-based Algorithm

Table IV contains the results of the MOS-based algorithm on the considered benchmark. From these results we can make the following observations:

- The algorithm is able to solve one function to the maximum precision (F_1) and another one to a very low error value (F_3).
- For all the functions the differences between the mean and the median are very low, which implies that our algorithm is rather robust.

C. Comparison with Composing Algorithms

Table V contains the statistical comparison of the MOS-based hybrid algorithm with each of its composing algorithms. For this comparison, we have used the well-known Wilcoxon signed-rank test. Additionally, and due to the limited size of the sample (the average error of just 15 functions), we have also included the standardized $p-value$ as described in [23]. The objective of this correction is to standardize the arbitrary $p-values$ for a sample size of 100 individuals and to correct the obtained $p-values$ according to the actual sample size (Eq. 2, where N is the sample size).

$$p_{stan} = min\left(\frac{1}{2}, p \cdot \frac{\sqrt{N}}{10}\right) \tag{2}$$

This correction tries to compensate the bias introduced by the sample size, making it easier to find significant differences when the sample size is smaller than 100 individuals and, consequently, making it harder when the sample size is over 100 individuals.

According to the Wilcoxon $p-value$, the MOS-based hybrid algorithm is statistically better than the GA and the Solis and Wets algorithms. It is also quite close to the 0.05 threshold when compared with the MTS-LS1-Reduced. If we take into account the standardized $p-values$, the hybrid algorithm is statistically better than any of its composing algorithms. We also report the overall $p-values$, taking into account the Family-Wise Error (FWER) [24]. Considering the raw Wilcoxon $p-value$, the MOS-based algorithm is close to be statistically better than its three composing algorithms globally, whereas the standardized $p-value$ reports significant results in this multiple comparison.

TABLE V. STATISTICAL VALIDATION (MOS IS THE CONTROL ALGORITHM)

MOS vs.	Wilcoxon p-value	Standardized p-value
MTS-LS1-Reduced	$6.03E-02$	$2.44E-02^{\surd}$
GA	$9.03E-03^{\surd}$	$3.50E-03^{\surd}$
Solis and Wets	$3.05E-05^{\surd}$	$1.18E-05^{\surd}$
Wilcoxon p-value with FWER: MOS vs. MTS-LS1-Reduced, GA, Solis and Wets	$9.06E-02$	$3.51E-02^{\surd}$

$\surd$ means that there are statistical differences with significance level $\alpha = 0.05$

D. Comparison with the Reference Algorithm

Table VI compares the results obtained by the MOS-based hybrid algorithm with those of the reference algorithm: DECC-GG [25]. In this table, the best results for each function appear with a light gray background to ease their further analysis. By comparing the results of both algorithms, we can observe the following:

- The MOS-based algorithm obtains the best performance in 14 out of 15 functions, whereas DECC-GG obtains the best results only in 1 function.
- In F_6, in which DECC-GG obtains better results, the difference with MOS is of just one order of magnitude. On the other hand, MOS is able to reduce the error in several orders of magnitude (up to four) in several functions.
- The Wilcoxon signed-rank test reports a $p-value$ of $3.05E-04$ ($1.18E-04$ for the standardized $p-value$), which means that the results obtained by the MOS-based algorithm are statistically better than those of DECC-G.

TABLE IV. EXPERIMENTAL RESULTS WITH MOS

		F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8
FEs = 1.2e+05	Best	1.13E+07	2.30E+03	6.31E+00	2.16E+10	5.25E+06	2.79E+05	1.46E+08	1.67E+14
	Median	2.99E+07	2.59E+03	7.77E+00	3.58E+10	6.80E+06	3.11E+05	3.28E+08	3.72E+14
	Worst	3.95E+07	3.12E+03	9.24E+00	4.41E+10	8.57E+06	3.91E+05	6.30E+08	5.49E+14
	Mean	2.71E+07	2.64E+03	7.85E+00	3.47E+10	6.96E+06	3.11E+05	3.46E+08	3.72E+14
	Std	7.97E+06	1.88E+02	6.53E-01	6.44E+09	9.09E+05	2.26E+04	1.39E+08	9.43E+13
FEs = 6.0e+05	Best	8.62E-02	1.63E+03	1.25E-11	1.39E+09	5.25E+06	1.82E+03	2.44E+06	4.13E+13
	Median	1.38E+00	1.77E+03	4.09E-11	2.46E+09	6.79E+06	1.39E+05	8.07E+06	8.56E+13
	Worst	1.80E+01	1.96E+03	1.22E-09	3.64E+09	8.56E+06	2.31E+05	1.53E+07	1.24E+14
	Mean	3.48E+00	1.78E+03	1.33E-10	2.56E+09	6.95E+06	1.48E+05	8.19E+06	8.41E+13
	Std	4.67E+00	8.24E+01	2.61E-10	5.14E+08	9.04E+05	6.54E+04	3.43E+06	2.23E+13
FEs = 3.0e+06	Best	0.00e+00	7.40e+02	8.20e-13	1.10e+08	5.25e+06	1.95e+01	3.49e+03	3.26e+12
	Median	0.00e+00	8.36e+02	9.10e-13	1.56e+08	6.79e+06	1.39e+05	1.62e+04	8.08e+12
	Worst	0.00e+00	9.28e+02	1.00e-12	5.22e+08	8.56e+06	2.31e+05	3.73e+04	1.32e+13
	Mean	0.00e+00	8.32e+02	9.17e-13	1.74e+08	6.94e+06	1.48e+05	1.62e+04	8.00e+12
	Std	0.00e+00	4.48e+01	5.12e-14	7.87e+07	8.85e+05	6.43e+04	9.10e+03	3.07e+12
		F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}	
FEs = 1.2e+05	Best	3.15E+08	7.46E+05	1.99E+09	7.27E+03	5.05E+09	1.88E+10	1.21E+07	
	Median	4.32E+08	1.24E+06	2.78E+09	1.02E+04	7.34E+09	4.46E+10	1.43E+07	
	Worst	6.01E+08	1.27E+06	7.00E+09	1.78E+04	1.38E+10	7.89E+10	1.83E+07	
	Mean	4.29E+08	1.16E+06	3.13E+09	1.16E+04	8.37E+09	4.61E+10	1.45E+07	
	Std	6.24E+07	1.70E+05	1.07E+09	3.61E+03	2.51E+09	1.71E+10	1.66E+06	
FEs = 6.0e+05	Best	2.64E+08	1.39E+03	4.94E+08	1.57E+03	3.32E+08	7.91E+07	5.32E+06	
	Median	3.89E+08	1.18E+06	7.79E+08	2.02E+03	7.64E+08	1.24E+08	6.25E+06	
	Worst	5.42E+08	1.23E+06	1.20E+09	5.54E+03	1.58E+09	1.70E+09	7.55E+06	
	Mean	3.84E+08	9.03E+05	8.05E+08	2.20E+03	8.10E+08	2.03E+08	6.24E+06	
	Std	6.40E+07	5.18E+05	1.60E+08	7.84E+02	2.62E+08	3.17E+08	5.97E+05	
FEs = 3.0e+06	Best	2.63e+08	5.92e+02	2.06e+07	2.22e-01	1.52e+06	1.54e+07	2.03e+06	
	Median	3.87e+08	1.18e+06	4.48e+07	2.46e+02	3.30e+06	2.42e+07	2.38e+06	
	Worst	5.42e+08	1.23e+06	9.50e+07	1.17e+03	6.16e+06	4.46e+07	2.88e+06	
	Mean	3.83e+08	9.02e+05	5.22e+07	2.47e+02	3.40e+06	2.56e+07	2.35e+06	
	Std	6.29e+07	5.07e+05	2.05e+07	2.54e+02	1.06e+06	7.94e+06	1.94e+05	

TABLE VI. COMPARISON WITH REFERENCE ALGORITHM, FES = 3.0E6

		F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8
DECC-CG	Best	1.57e-13	9.90e+02	2.63e-10	7.58e+09	7.28e+14	6.96e-08	1.96e+08	1.43e+14
	Median	2.00e-13	1.03e+03	2.85e-10	2.12e+10	7.28e+14	6.08e+04	4.27e+08	3.88e+14
	Worst	2.45e-13	1.07e+03	3.16e-10	6.99e+10	7.28e+14	1.10e+05	1.78e+09	7.75e+14
	Mean	2.03e-13	1.03e+03	2.87e-10	2.60e+10	7.28e+14	4.85e+04	6.07e+08	4.26e+14
	Std	1.78e-14	2.26e+01	1.38e-11	1.47e+10	1.51e+05	3.98e+04	4.09e+08	1.53e+14
MOS	Best	0.00e+00	7.40e+02	8.20e-13	1.10e+08	5.25e+06	1.95e+01	3.49e+03	3.26e+12
	Median	0.00e+00	8.36e+02	9.10e-13	1.56e+08	6.79e+06	1.39e+05	1.62e+04	8.08e+12
	Worst	0.00e+00	9.28e+02	1.00e-12	5.22e+08	8.56e+06	2.31e+05	3.73e+04	1.32e+13
	Mean	0.00e+00	8.32e+02	9.17e-13	1.74e+08	6.94e+06	1.48e+05	1.62e+04	8.00e+12
	Std	0.00e+00	4.48e+01	5.12e-14	7.87e+07	8.85e+05	6.43e+04	9.10e+03	3.07e+12
		F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}	
DECC-CG	Best	2.20e+08	9.29e+04	4.68e+10	9.80e+02	2.09e+10	1.91e+11	4.63e+07	
	Median	4.17e+08	1.19e+07	1.60e+11	1.03e+03	3.36e+10	6.27e+11	6.01e+07	
	Worst	6.55e+08	1.73e+07	7.16e+11	1.20e+03	4.64e+10	1.04e+12	7.15e+07	
	Mean	4.27e+08	1.10e+07	2.46e+11	1.04e+03	3.42e+10	6.08e+11	6.05e+07	
	Std	9.89e+07	4.00e+06	2.03e+11	5.76e+01	6.41e+09	2.06e+11	6.45e+06	
MOS	Best	2.63e+08	5.92e+02	2.06e+07	2.22e-01	1.52e+06	1.54e+07	2.03e+06	
	Median	3.87e+08	1.18e+06	4.48e+07	2.46e+02	3.30e+06	2.42e+07	2.38e+06	
	Worst	5.42e+08	1.23e+06	9.50e+07	1.17e+03	6.16e+06	4.46e+07	2.88e+06	
	Mean	3.83e+08	9.02e+05	5.22e+07	2.47e+02	3.40e+06	2.56e+07	2.35e+06	
	Std	6.29e+07	5.07e+05	2.05e+07	2.54e+02	1.06e+06	7.94e+06	1.94e+05	

VI. Conclusions

In this paper we have tested several well-known algorithms, as well as a new variant of the first of the local searches of the MTS algorithm, which we have named MTS-LS1-Reduced, and a new DE variant combining the principles of the self-adaptation of the parameters of DE and the opposition search on a set of 15 large scale continuous functions with an experimental design defined with the Taguchi method. This experimentation allowed us to find the best configuration of parameters for each of the individual algorithms and compare them with statistical tests to select which of them should be combined. Then, a similar procedure has been conducted to create the hybrid algorithm combining a Genetic Algorithm, the Solis and Wets algorithm and the MTS-LS1-Reduced, within the MOS framework. The best hybrid configuration was compared with its composing algorithms as well as with the well-known DECC-GG algorithm. This experimentation shows that our proposal obtains significantly better results than any of its composing algorithms and also than the reference algorithm. In a future work, the parameters study could be extended as we conjecture that there is still room for improvement on the performance of the hybrid algorithm. For example, the GA may work better with a larger population size. Furthermore, a mechanism to combine several population-based algorithms with different population sizes needs will also be studied, as preliminary efforts do not yield satisfactory results.

Acknowledgments

This work was financed by the Spanish Ministry of Science (TIN2010-21289-C02-02) and supported by the Cajal Blue Brain Project. The authors thankfully acknowledge the computer resources, technical expertise and assistance provided by the Centro de Supercomputación y Visualización de Madrid (CeSViMa) and the Spanish Supercomputing Network. A. LaTorre gratefully acknowledges the support of the Spanish Ministry of Science and Innovation (MICINN) for its funding throughout the Juan de la Cierva program.

References

[1] T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, Dec. 1995.

[2] L. Y. Tseng and C. Chen, "Multiple Trajectory Search for Large Scale Global Optimization," in *Proceedings of the 10th IEEE Congress on Evolutionary Computation, CEC 2008 (IEEE World Congress on Computational Intelligence)*. IEEE Press, Jun. 2008, pp. 3052–3059.

[3] A. LaTorre, S. Muelas, and J. M. Peña, "A MOS-based dynamic memetic differential evolution algorithm for continuous optimization: a scalability test," *Soft Computing-A Fusion of Foundations*, vol. 15, no. 11, pp. 2187–2199, 2011.

[4] X. Li, K. Tang, M. N. Omidvar, Z. Yang, and K. Qin, "Benchmark Functions for the CEC'2013 Special Session and Competition on Large-Scale Global Optimization," Tech. Rep., 2013.

[5] E.-G. Talbi, "A Taxonomy of Hybrid Metaheuristics," *Journal of Heuristics*, vol. 8, no. 5, pp. 541–564, Sep. 2002.

[6] D. Molina, M. Lozano, and F. Herrera, "MA-SW-Chains: Memetic Algorithm Based on Local Search Chains for Large Scale Continuous Global Optimization," *Proceedings of the 2010 IEEE Congress on Evolutionary Computation, CEC 2010*, pp. 1–8, 2010.

[7] M. Lozano, D. Molina, and F. Herrera, "Editorial Scalability of Evolutionary Algorithms and Other Metaheuristics for Large-Scale Continuous Optimization Problems," *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, vol. 15, pp. 2085–2087, 2011.

[8] A. LaTorre, S. Muelas, and J. M. Peña, "Multiple Offspring Sampling in Large Scale Global Optimization," in *Proceedings of the 2012 IEEE Congress on Evolutionary Computation*, Brisbane, Australia, Jun. 2012, pp. 964–971.

[9] Y. Chen, Y. S. Ong, M. H. Lim, and K. C. Tan, "A Multi-Facet Survey on Memetic Computation," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 5, pp. 591–607, 2011.

[10] F. Neri, C. Cotta, and P. Moscato, Eds., *Handbook of Memetic Algorithms*, ser. Studies in Computational Intelligence. Springer, 2012, vol. 379.

[11] F. Herrera and M. Lozano, "Gradual Distributed Real-Coded Genetic Algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 1, pp. 43–63, 2000.

[12] R. Storn and K. V. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[13] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-Adapting Control Parameters in Differential Evolution: a Comparative Study on Numerical Benchmark Problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006.

[14] H. Wang, Z. Wu, S. Rahnamayan, and L. Kang, "A Scalability Test for Accelerated DE Using Generalized Opposition-Based Learning," in *Proceedings of the 9th International Conference on Intelligent Systems Design and Applications, ISDA 2009*, pp. 1090–1095.

[15] F. J. Solis and R. J. B. Wets, "Minimization by Random Search Techniques," *Mathematics of Operations Research*, vol. 6, no. 1, pp. 19–30, Feb. 1981.

[16] A. LaTorre, "A Framework for Hybrid Dynamic Evolutionary Algorithms: Multiple Offspring Sampling (MOS)," Ph.D. dissertation, Universidad Politécnica de Madrid, Nov. 2009.

[17] A. LaTorre, S. Muelas, and J. M. Peña, "Benchmarking a MOS-based algorithm on the BBOB-2010 Noiseless Function Testbed," in *12th Genetic and Evolutionary Computation Conference, GECCO 2010 (Companion)*. ACM, 2010, pp. 1649–1656.

[18] A. LaTorre, J. M. Peña, S. Muelas, and M. Zaforas, "Hybrid Evolutionary Algorithms for Large Scale Continuous Problems," in *11th Genetic and Evolutionary Computation Conference, GECCO 2009*. ACM Press, 2009, pp. 1863–1865.

[19] G. Taguchi, S. Chowdhury, and Y. Wu, *Taguchi's Quality Engineering Handbook*. John Wiley.

[20] M. Friedman, "The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance," *Journal of the American Statistical Association*, vol. 32, no. 200, pp. 675–701, 1937.

[21] S. Muelas, J. M. Peña, V. Robles, A. LaTorre, and P. de Miguel, "Machine Learning Methods to Analyze Migration Parameters in Parallel Genetic Algorithms," in *Proceedings of the International Workshop on Hybrid Artificial Intelligence Systems 2007*, E. Corchado, J. M. Corchado, and A. Abraham, Eds. Salamanca, Spain: Springer Verlag, Nov. 2007, pp. 199–206.

[22] F. Wilcoxon, "Individual Comparisons by Ranking Methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[23] I. J. Good, "The Bayes/Non-Bayes Compromise: a Brief Review," *Journal of the American Statistical Association*, vol. 87, no. 419, pp. 597–606, 1992.

[24] S. García, D. Molina, M. Lozano, and F. Herrera, "A Study on the Use of Non-Parametric Tests for Analyzing the Evolutionary Algorithms' Behaviour: A Case Study on the CEC'2005 Special Session on Real Parameter Optimization," *Journal of Heuristics*, vol. 15, no. 6, pp. 617–644, Dec. 2009.

[25] Z. Yang, K. Tang, and X. Yao, "Large Scale Evolutionary Optimization Using Cooperative Coevolution," *Information Sciences*, vol. 178, no. 15, pp. 2985–2999, 2008.