Comparing Large-Scale Global Optimization Competition winners in a real-world problem

1st Daniel Molina
Andalusian Institute of Data Science (DASCI)
and Computational Intelligence Dept.
University of Granada
Granada, Spain
dmolina@decsai.ugr.es

2nd Arthur R. Nesterenko
University of Granada
Granada, Spain
arthur18@correo.ugr.es

3rd Antonio LaTorre
Center for Computational Simulation (CCS)
Universidad Politécnica de Madrid
Madrid, Spain
a.latorre@upm.es

***Abstract*—The optimization of thousands of variables, Large-Scale Global Optimization, is a research topic that is obtaining more and more attention by its applications in engineering and medical problems. In order to design evolutionary algorithms for these problems, several specific competitions have been organized, using benchmarks such as the ones proposed in CEC'2010 and CEC'2013, trying to simulate realistic features of real-world problems. Several algorithms have been proposed, some of them being very competitive on these benchmarks, especially during the last years. However, all of them were tested only on those artificial benchmarks, so there are no guarantees that they would obtain good performance in more realistic problems. In this paper, we select the best algorithms in these competitions to optimize a real-world problem, an electroencephalography (EEG) optimization problem. The new benchmark contains noisy problems and an increasing number of variables (up to 5000) compared to synthetic benchmarks (limited to 1000 variables). Results show that, although the fitness obtained by the majority of the algorithms is the same, the processing time strongly depends on the algorithm under consideration. The optimization time for a fixed number of fitness evaluations varies, in the most complex problems, from 3 hours to around 18 minutes, being MOS-2013 the fastest algorithm. However, if we focus our attention on the time needed to reach the best-known solution, SHADEILS becomes the fastest algorithm (with a maximum of three minutes). In our opinion, this should encourage researchers to continue working in more scalable and efficient algorithms for large-scale global optimization.**

***Index Terms*—Large-scale global optimization, LSGO, Benchmarking, BComp.**

I. Introduction

The number of decision variables in engineering problems has grown exponentially during the last 50 years [1], and this trend has increased over the years. Nowadays, in certain areas, many real-world optimization problems involve optimizing a large number of decision variables, hundreds, if not thousands, of continuous parameters. This type of optimization problems, with thousands of variables, is called Large-Scale Global Optimization (LSGO).

Evolutionary Algorithms [2] are a very popular tool in the field of real coding optimization, in industrial and scientific domains. However, the larger the number of variables to optimize, the worse the behavior of these algorithms, as the domain search increases exponentially with the dimension (the *curse of dimensionality* [3]).

During the last years, several algorithms especially designed for LSGO have been proposed [4], and evaluated on specific benchmarks for LSGO such as [5], [6]. These benchmarks try to simulate some of the characteristics of real-world problems, such as, the heterogeneous contribution of problem variables to the fitness function.

However, these algorithms have not been tested against a different test suite, to observe if their competitiveness on synthetic benchmarks can be extrapolated to real-world problems, and the potential challenges to apply them in this scenarios.

In this paper, we are going to compare the different winners of some recent LSGO competitions against a real-world problem, a big electroencephalography (EEG) data optimization [7]. This is a very interesting benchmark because it is a real-world problem, with both noisy and noiseless versions, and with instances with different number of variables, much bigger in its largest configuration than the other benchmarks.

This work has the following structure: In Section II, the new benchmark is described in detail, remarking the differences with previous synthetic benchmarks. In Section III the different algorithms under comparison are briefly described. In Section IV, the results obtained by the different reference algorithms are analyzed and discussed. Finally, in Section V main conclusions and some future work are summarized.

II. Big Optimization Benchmark

Electroencephalography (EEG) is a valuable tool for research and diagnosis. It is an electrophysiological monitoring method to record electrical activity of the brain in a noninvasive way, with the electrodes placed along the scalp. EEG is most often used to diagnose epilepsy, and also to diagnose sleep disorders.

One important drawback of EEG is the distortion due to non-brain signals, called artifacts, tarnishing the results obtained. A visual inspection by a human can remove the artifacts influence over the obtained components, recomposing the original EEG, but this manual process is very time-consuming.

In 2015, a new benchmark for the Big Data Competition was proposed [1] [7]. This benchmark is made up of three sub-

[1]http://www.husseinabbass.net/BigOpt.html

problems, which differ only in the number of electrodes (and, therefore, in the number of variables to be optimized). The three datasets: A, B, and C, have both a noisy and a noiseless version.

To summarize, there are 6 problems, identified by the number of electrodes and a N indicating whether it is the noisy or the noiseless version: D4, D4N, D12, D12N, D19 and D19N. As each signal is sampled using 256Hz, the number of variables is 1024, 3072, and 4864 respectively, as shown in Table I.

Dataset	Number electrodes	Artifacts	Variables
A	4	2	1024
B	12	6	3072
C	19	6	4864

TABLE I: Subproblems from BigOpt benchmark

The objective is to apply Independent Component Analysis, ICA, to minimize the influence of the artifacts to improve the signals. Mathematically, it implies to calculate the signal S that satisfies:

$$X = A \cdot S + N \tag{1}$$

where X is the obtained signal, N is the noise, and A is a lineal transformation matrix.

The problem is to decompose S in S_1 and S_2 such that $S = S_1 + S_2$ and $X = A \times S$. Because S_2 can be calculated from S_1, the goal is to obtain the S_1 matrix that optimizes the following two objectives:

1) Maximize the Pearson correlation, reducing f_1:

$$f_1 = \frac{1}{N^2 - N} \sum_i \sum_{j \neq i} C_{ij} + \frac{1}{N} \sum_i (1 - C_{ii})^2 \tag{2}$$

where

$$C = \frac{covar(X, A \cdot S1)}{\sigma(X) \cdot \sigma(A \cdot S_1)}$$

2) Reduce the distance between S and S_1:

$$f_2 = \frac{1}{N \times M} \sum_i \sum_j (S_{ij} - S_{1_{ij}})^2 \tag{3}$$

Given that most of the algorithms proposed for LSGO are for only a single objective, the fitness function has been converted into its mono-objective version, in which the actual fitness function being optimized is a lineal combination of the two objectives: $fitness = Minimize(f_1 + f_2)$.

The data files for each of the 6 problems were generated with synthetic sampling, and are freely available. Unfortunately, the source code of the fitness function is not available: only a description of how f_1 and f_2 should be calculated is provided.

This benchmark has the following advantages:

- It simulates a real-world problem, with clear interest.
- It provides different datasets with different number of variables, several times bigger than those used in classic LSGO benchmarks.
- Noisy and noiseless problems are available. It is common in real-world competitions, but not in LSGO ones.

However, it has also several important drawbacks:

- It is not a full real-world problem, because the data are synthetically generated, not coming from real patients.
- The implementation was not publicly available, and we had to write multiple times to authors of the benchmark. It is very important to be sure that differences in results are due to the algorithms, and not to implementations of the benchmark.
- Few algorithms have reported results for this benchmark. To the best of our knowledge, only the following three [7], [8], [9]. Maybe the reason for these few reference algorithms after 3 years is the absence of a publicly available implementation, as stated before.

III. Previous winner algorithms in LSGO

In this section we briefly describe the algorithms used as reference. All of them have been selected by their good results in previous LSGO benchmarks:

- MOS version 2011 [10], winner of the SOCO 2011 Special Issue, with a benchmark specifically designed for this competition.
- MOS version 2013 [11], winner in the CEC'2013 competition with the CEC'2013 LSGO benchmark. It was unbeaten until the competition of the IEEE CEC'2018.
- SHADEILS [12], winner in the CEC'2018 competition with the CEC'2013 LSGO benchmark.
- MLSHADE-SPA [13], runner-up in the CEC'2018 competition, with better results than MOS version 2013.

In the following subsections, we are going to briefly describe these four algorithms.

A. Multiple Offspring Sampling (version 2011)

This was the winner of the SOCO 2011 Special Issue. It is a hybrid algorithm that combines a differential evolution algorithm (DE) and a LS method specially designed for LSGO, MTS-LS1. MTS-LS1 tries to optimize each variable independently by moving them in different directions. It uses a step size that is adapted when no improvement is possible with the current value. It is initialized with a large value to favor exploration and is systematically adjusted to better search in the proximity of an optimum.

The most characteristic element of MOS is the division of the allocated budget of fitness evaluations in several steps and the use of a participation function and a quality measure to decide the budget, in terms of fitness evaluations, that each of its composing techniques is allowed to use at each step. The quality measure evaluates how well a particular subcomponent is performing and then the participation function adjusts the budgets accordingly. This behavior is common to the 2013 version. However, the individual components used in each case differ.

If we analyze the two components being used in MOS-2011, it is interesting to note how one of the two techniques is a classic algorithm in its most basic configuration (DE), which has not been specifically designed for LSGO. However, its combination with a powerful local search (MTS-LS1) yields excellent reports, still not beaten for this benchmark.

Details on the implementation and the parameters of the algorithm can be obtained from [10].

B. Multiple Offspring Sampling (version 2013)

In the CEC'2013 competition a new CEC'2013 benchmark was proposed. In that year, a new version of MOS was proposed, obtaining the best results for the following five years. This version maintains the general dynamic framework described for MOS-2011. However, the individual components used by the algorithm are different. In this case, a simple Genetic Algorithm and two local searches, an evolution of MTS-LS1 called MTS-LS1-Red and the classic Solis-Wets method, are used. Regarding the improved MTS-LS1 algorithm (MTS-LS1-Red), its main difference is that it stores the improvements in the objective value associated to each variable of the problem, focusing the search more on those components than on others (remember that the original MTS-LS1 searched all the components) with the same effort.

In [11] all the details about the algorithm are provided and results are compared with a number of state-of-the-art LSGO algorithms.

C. SHADE with Iterative Local Search

SHADE with Iterative Local Search (SHADE-ILS) is an algorithm that iterative applies a well-known DE variant, SHADE, in combination with a LS component: a step of the SHADE algorithm of Max_{DE} evaluations, is followed by one of the LS method of the same duration.

The main characteristics of SHADE-ILS are:

- SHADE is an advanced DE with self-adaptation of the DE parameters. Additionally, during mutation it uses an archive of previous solutions (to increase diversity) and it is guided by the p best solutions (to exploit best solutions).
- At each iteration of the LS method, the algorithm chooses one of the following available techniques: the MTS-LS1 (used also by MOS-2011), and the classic L-BFGS-B [14] that uses an approximation of the gradient to improve the search.
- The selection of the LS is carried out considering the relative improvement ratio during its last application.

More details on the design of the algorithm, its configuration and a comparative with other state-of-the-art LSGO methods can be found in [12].

D. L-SHADE Memetic framework with semi-parameter adaptation

LSHADE-SPA Memetic Framework with semi-parameter adaptation (MLSHADE-SPA) is a hybridization framework between population-based algorithms and local search methods. The Cooperative Coevolution framework (CC) manages a total of four techniques:

- Three DE algorithms used for exploration purposes: LSHADE-SPA (Semi-parameter adaptation SHADE with linear-population reduction), EADE (DE with random triplex-vector mutation operator) and ANDE (another DE with random convex triplex-vector mutation operator).
- A modified version of the MTS algorithm (MMTS), which selects the starting explotation point from the input population after the exploration process.

The main difference with MOS is the use of the CC framework, by which the algorithm randomly divides the set of variables in several groups, and each group is optimized by each of the optimization components. In addition to that, the population during the application of the components can increase. In that case, only the best solutions are kept into the population (to maintain stable the population size). You can check all the details of the algorithm and its parameters in [13].

IV. Experimental Section

This section describes the experimental setup, with details of the parameters values of the different algorithms under consideration, a description of the system used to carry out the experiments and of the experimental procedure followed. Moreover, a comparison and analysis of the results observed is also provided.

Regarding the implementation of the different algorithms, we have used the original implementations as provided by their authors, with minimum changes to include the fitness function of the BigOpt benchmark. Furthermore, the source code of the test suite was also obtained from its authors. This was very important to us to avoid, as discussed before, the risk of any possible error in the implementation of the fitness function that could alter the results.

As the source code of all the algorithms was retrieved from their original authors, several programming languages have been used:

- Both versions of MOS (MOS-2011 and MOS-2013) were implemented in C++.
- SHADEILS was implemented in Python3, using the Numpy library for getting better performance.
- MLSHADE-SPA was implemented in Matlab.

All the experiments were run using the same hardware configuration, to allow fair comparisons of running times. In particular, experiments were run on a cluster with nodes with the following configuration: Intel Core i7 930 processors at 2.8Ghz; Ubuntu 18.04 LTS Operating System and 24GB of RAM. The C++ Compiler was gcc 7.3.0.; Python interpreter was 3.6.4 and MATLAB version was 9.4 (R2018a). Each algorithm was run sequentially (neither MPI or other parallelism technique was used), the different nodes were used to run in parallel over the different problems.

Each algorithm was run for $1,000,000$ fitness evaluations for each problem, 10 times, and the average results were obtained.

Finally, for the parameter of the different algorithms we used the recommended values by their authors, as shown in Tables II-V. We decided to keep these values because it is usually the followed criterion in a real-world problem. The idea is to test each algorithm as it was proposed by their authors on a new problem, without any parameter tuning.

Parameter	Value	Parameter	Value
Population size	15	Step size	35715
DE F	0.5	DE CR	0.5

TABLE II: Parameter values for MOS-2011

Context	Parameter	Value	Parameter	Value
Global	Population size	400	Crossover PCX	0.9
	Prob mutation	0.01	Minimum	20%
	Step size	3600	Selection	Tournament
Solis Wets	adjustSuccess	4	adjustFailed	0.75
	maxSuccess	5	maxFailed	4
	delta	2.4		
MTS-LS-Red	adjustFailed	2	adjustMin	10
	moveLeft	0.25	moveRight	0.5
	searchProb	90%	minProb	2.5%

TABLE III: Parameter values for MOS-2013

Parameter	Value	Parameter	Value
Population size	100	$Freq_{LS}$	10
FE_{DE}	25000	FE_{LS}	25000
Threshold	1%	Times without imp.	3
MTS initial step	20		

TABLE IV: Parameter values for SHADEILS

Parameter	Value	Parameter	Value
Initial popsize	250	Minimum popsize	20
Memory size	5	Evals CC	50
Ratio Pbest	0.1		

TABLE V: Parameter values for MLSHADE-SPA

We are going to start our analysis by comparing the fitness obtained by each of the algorithms used in this studio, plus an additional one, MAGA [9], an algorithm specially designed for that benchmark that obtained the best results. Table VI shows the average fitness for each of the compared algorithms, and also for the reference one.

From Table VI, we can obtain the following conclusions:

- The results obtained by the different algorithms are surprisingly similar. Only MOS-2011 achieves worse results than MAGA, the previous reference algorithm. All the other algorithms obtain very similar results (probably, observed differences are due to the number of significant digits used by each implementation).
- It can be observed that the noise in the datasets has very little influence on the results. Moreover, for several problems, the noisy version is easier to optimize than the noiseless one).
- Considering the different problems, the D4 problem (1024 variables) may seem the simpler one to solve *a priori*. However, what really happens is that less information is available - a smaller amount of mixed signals - of which the algorithm needs to identify the correlation between the signals and separate the artifacts from them. Also, it is the least representative of a real environment, because in real situations it is common to find more electrodes. In the D12 problem (3072 variables) algorithms achieve the best results because, even if it has more variables to optimize, there is more information available to do it. D19 is the problem with the highest difficulty due to the highest domain search, and thus it was expected to get worse results than with D12, but it still can be optimized better than D4. To summarize this point, even when dimensionality implies more complex landscapes, in real-world problems, sometimes a certain number of variables could be required to obtain enough information to be optimized.
- Even if the algorithms compared in this studio (except for the case of MAGA) have been designed for 1000 dimensions benchmark, they have shown a very robust behavior for a much larger benchmark, reaching the best results for the 3000 variables problems.

Our next analysis will consider algorithmic complexity, in terms of running time. We are going to analyse how the runtime of each algorithm depends on the number of variables (D4 implies 1000 variables, a similar problem size to those used in the CEC'2013 benchmark, but D12 already implies 3000 variables, and D19 4800 variables, much larger problems), by using only one computer/node (with the same configuration for all the algorithms, to allow a fair comparison).

Table VII presents the time for a single run of each algorithm. It can be seen that, whereas almost no differences were reported in fitness values, processing time shows big differences among the algorithms, with running times ranging from more than 3 hours for the slowest algorithm, MLSHADE-SPA, to the 18 minutes of the fastest one, MOS-CEC2013 (these times correspond to the D19 problems). Figure 1 graphically depicts these differences in computing time for the algorithms under consideration. It is obvious that MLSHADE-SPA is less scalable than the other algorithms, whereas MOS-2013 seems to be the fastest method. SHADEILS is slower than MOS-based algorithms, but these differences decrease as dimensionality increases, obtaining, for problem D19, very similar running times to those of MOS-2011.

In order to highlight these time differences, Table VIII shows the relative time of each algorithm to the slowest one. The following conclusions can be obtained:

- MLSHADE-SPA is clearly the algorithm with worst execution time. In the smallest problem, it already takes twice the time of the next algorithm, and these differences increase with dimensionality.

Problem	MOS-2011	MOS-2013	SHADE-ILS	MLSHADE-SPA	MAGA*
D4	0.06103	0.06103	0.06103	0.06103	0.0610
D4N	0.05897	0.05897	0.05897	0.05897	0.0590
D12	0.00198	0.00194	0.00194	0.00194	0.0019
D12N	0.00188	0.00183	0.00183	0.00183	0.0018
D19	0.0894	0.00251	0.00252	0.00252	0.0025
D19N	0.0918	0.00256	0.00256	0.00256	0.0026

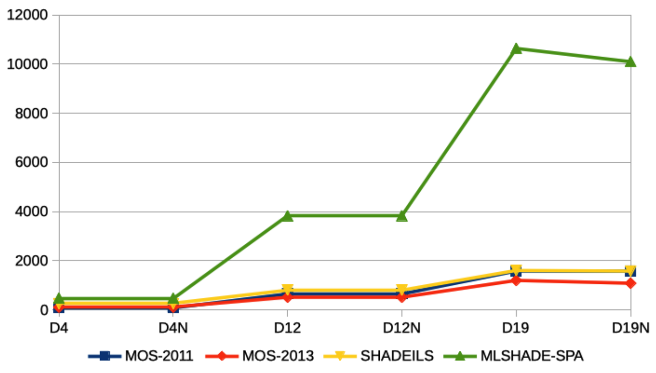
TABLE VI: Average fitness for each algorithm and problem

Algorithm	D4	D4N	D12	D12N	D19	D19N
MOS-2011	98	98	660	661	1582	1578
MOS-2013	111	111	519	518	1202	1083
SHADEILS	256	257	811	804	1604	1580
MLSHADE-SPA	474	469	3825	3831	10637	10096

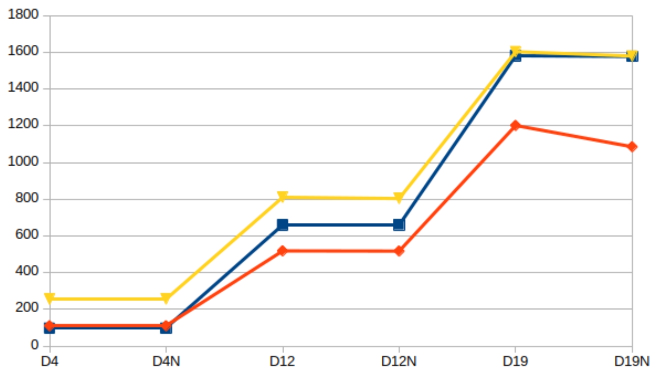
TABLE VII: Average time, in seconds, after 10 runs

Algorithm	D4	D4N	D12	D12N	D19	D19N
MOS-2011	19	19	16	16	13	13
MOS-2013	20	20	13	12	10	10
SHADEILS	51	49	20	20	14	14
MLSHADE-SPA	100	100	100	100	100	100

TABLE VIII: Relative time to the slowest algorithm



(a) All considered algorithms



(b) Fastest algorithms, to highlight time differences

Fig. 1: Running time for each algorithm and problem

- MOS-2013 not only improves the results of MOS-2011, but is also faster. For dimension 1000 it is slighly slower, but in more complex problems it uses less time than MOS-2011, and the difference increases with dimension (for D19, MOS-2013 uses 25% less time).
- SHADEILS is the second slowest algorithm. However, in higher dimensions, the differences decrease, taking a very similar time to solve D19 to that of MOS-2011 (and obtains better results). Thus, it is a rather scalable algorithm.

These results have a strong relationship with the programming language/technology used for the implementation of the different algorithms: MOS is programmed in C++, one of the fastest existing programming languages. MLSHADE-SPA is coded in Matlab, and it is expected to take more time than a C++ implementation. Finally, SHADEILS is implemented in Python and, even if it uses the well-known, and optimized, Numpy library, we could expect it to be slighly slower than Matlab, and several times slower than a C++ implementation. However, in this case, as SHADEILS is more scalable, it is a very competitive algorithm (with similar times to those of MOS-2011).

In the previous paragraphs we have compared the running time of the algorithms with a fixed number of evaluation. However, in real-world problems, the important time is sometimes the required one to obtain a competitive solution. In order to study that, we have tested the number of evaluations required to achieve the current best known solution for each algorithm.

Table IX shows the results for the time required for each algorithm to achieve the best known solutions (in MOS-2011, they were not always obtained). The data in the table corresponds to the average time for 10 runs. Moreover, Table X shows the ratio against the total time used to obtain them.

Algorithm	D4	D4N	D12	D12N	D19	D19N
MOS-2011	14.8	14.4	591.4	624.3	-	-
MOS-2013	7.3	7.4	146.2	145.4	434.9	381.5
SHADEILS	1.4	1.2	120.2	116.5	187.9	184.2
MLSHADE-SPA	132.0	129.2	1565.0	1592.1	4591.4	4162.9

TABLE IX: Time, in seconds, required to achieve the best solution (optimum) for each algorithm and problem

From the results presented in the previous tables, it can be observed that:

- First, the maximum number of evaluations was excessive for the majority of the algorithms (with the exception of MOS-2011). Several algorithms reached their best

Algorithm	D4	D4N	D12	D12N	D19	D19N
MOS-2011	15.10	14.69	89.61	94.45	-	-
MOS-2013	6.58	6.67	28.17	28.07	36.18	35.23
SHADEILS	0.53	0.47	14.82	14.49	11.71	11.66
MLSHADE-SPA	27.84	27.50	40.91	41.55	43.16	41.22

TABLE X: Ratio of time in which each algorithm achieved the best known solution

solutions in a small fraction of the overall time (35% in MOS-2013, and 11% in SHADEILS).

- MLSHADE-SPA was the slowest algorithm. Not only it took more time for a fixed number of evaluations, but also it required more time to achieve the optima. For lower dimensions, it needs more than 25% of the overall time, and for higher dimensions, it requires more than 40% of the total time (for D19 that means around 70 minutes, more than ten times more than the rest of algorithms).
- MOS-2011 not always found the optima and, when it does, it takes more time than MOS-2013 (twice as much time for D4, and 4 times more in D12). Additionally, in D12 it needs the 95% of the overall time to reach them.
- MOS-2013 is able to obtain the best solutions in only 35% of the total time, in only 6 minutes instead of the aforementioned 18 minutes.
- In the case of SHADEILS, although it consumes more time than MOS-2013 for the same number of evaluations, it is able to achieve the optima in a smaller fraction of them (with only an 11% of the time it was able to solve all the problems). SHADEILS needed only 3 minutes to solve D19, whereas MOS-2013 needs 6 minutes.

To summarize, while the majority of the algorithms were able to achieve the same fitness for the problems in the new benchmark, there is a lot of difference in the computing time required, as can be seen in Figure 1. Furthermore, the required time to obtain the best known solution for several of them was only a small fraction of time. MOS-2013 and SHADEILS were clearly the most competitive algorithms. MOS-2013 runs faster than SHADEILS with the same number of evaluations (18 minutes vs 25 minutes), but SHADEILS was able to achieve the best known solutions in less time (3 minutes vs 6 minutes). Also, SHADEILS presents a very scalable behavior when the number of variables increases.

V. Conclusions

In this paper, we have compared the best algorithms in previous LSGO competitions, that used artificial benchmarks such as the ones proposed in CEC'2010 and CEC'2013, with a more realistic problem. The idea was to test if competitive algorithms using an artificial benchmark could achieve good results in this new problem without modifications.

The selected problem was an electrophysiological (EEG) problem, in which the function to optimize is a decomposition of the measured signals to reduce the distortions due to non-brain signals. The size of the instances depend on the number of electrodes used, ranging from 1000 variables (the same as in artificial problems) to 3100 and close to 5000 variables. Thus, this work can also be seen as a study of the scalability of the selected algorithms. The compared algorithms were MOS, in versions proposed in 2011 and 2013, and the algorithms that surpassed MOS-2013 in the CEC'2018 competition, MLSHADE-SPA and SHADEILS.

The experimental results have shown that, while the majority of the algorithms obtained a very similar fitness, there were a lot of differences in the required time to run each of them on the higher dimensionality problems (maintaining the same number of evaluations). The slowest algorithm took 3 hours to execute for the largest problem, whereas the fastest one, MOS-2013, took 18 minutes for the same problem size. These results are due to the efficiency of algorithms, and the technology used in their implementation. Measuring the time required to obtain the best known solutions, it was observed that several algorithms could achieve them with a small ratio of the overall optimization time (35% in MOS-2013, 11% in SHADEILS). From that perspective, the resulting fastest algorithm was SHADEILS, because it was able to get the optima in only 3 minutes, whereas MOS-2013 required twice as much time: 6 minutes. This shows that the efficiency of algorithms was much more influential than the technology/language used to implement them.

To conclude, experimental results show that the scalability of algorithms is an important factor that should be better studied in the LSGO competitions. In real-world optimization problems, processing time is determinant to decide if a problem is feasible, and thus the algorithms in LSGO should continue improving performance (not only the fitness) to improve their application to real-world LSGO problems.

As a future work, we are going to implement the promising SHADEILS in a compiled language, to obtain even more performance, and also to make it easier to use in real-world problems. Finally, we will study parallel versions of these LSGO algorithms, to decrease the required time for this kind of problems and to be able to tackle even larger ones.

Acknowledgments

This work was supported by grants from the Spanish Ministry of Science (TIN2014-57481-C2-2-R, TIN2016-8113-R, TIN2017-83132-C2-2-R and TIN2017-89517-P) and Universidad Politécnica de Madrid (PINV-18-XEOGHQ-19-4QTEBP).

References

[1] G. N. Vanderplaats, *Very large scale optimization.* National Aeronautics and Space Administration (NASA), Langley Research Center, 2002.

[2] T. Bäck, D. B. Fogel, and Z. Michalewicz, Eds., *Handbook of Evolutionary Computation.* Bristol, UK: IOP Publishing Ltd., 1997.

[3] S. Chen, J. Montgomery, and A. B. Röhler, "Measuring the curse of dimensionality and its effects on particle swarm optimization and differential evolution," *Applied Intelligence*, vol. 42, no. 3, pp. 514–526, 2015.

[4] D. Molina, A. LaTorre, and F. Herrera, "An insight into bio-inspired and evolutionary algorithms for global optimization: Review, analysis, and lessons learnt over a decade of competitions," *Cognitive Computation*, vol. 10, no. 4, pp. 517–544, Aug 2018. [Online]. Available: https://doi.org/10.1007/s12559-018-9554-0

[5] K. Tang, X. Li, P. Suganthan, Z. Yang, and T. Weise, “Benchmark Functions for the CEC’2010 Special Session and Competition on Large Scale Global Optimization,” Tech. Rep., 2010.

[6] 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,” RMIT University, Melbourne, Australia, Technical Report, 2013.

[7] S. K. Goh, K. C. Tan, A. A. Mamun, and H. A. Abbass, “Evolutionary big optimization (bigopt) of signals,” in *IEEE Congress on Evolutionary Computation, CEC 2015, Sendai, Japan, May 25-28, 2015*, 2015, pp. 3332–3339.

[8] Y. Zhang, M. Zhou, Z. Jiang, and J. Liu, “A multi-agent genetic algorithm for big optimization problems,” in *2015 IEEE Congress on Evolutionary Computation (CEC)*, May 2015, pp. 703–707.

[9] M. N. Omidvar, X. Li, Y. Mei, and X. Yao, “Cooperative co-evolution with differential grouping for large scale optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 3, pp. 378–393, June 2014.

[10] A. LaTorre, S. Muelas, and J. M. Peña, “A mos-based dynamic memetic differential evolution algorithm for continuous optimization: a scalability test,” Nov 2011, pp. 2187–2199.

[11] ——, “Large Scale Global Optimization: Experimental Results with MOS-based Hybrid Algorithms,” in *2013 IEEE Congress on Evolutionary Computation (CEC 2013)*, Cancún, Mexico, 2013, pp. 2742–2749.

[12] D. Molina, A. LaTorre, and F. Herrera, “SHADE with iterative local search for large-scale global optimization,” in *2018 IEEE Congress on Evolutionary Computation, CEC 2018, Rio de Janeiro, Brazil, July 8-13, 2018*, 2018, pp. 1252–1259.

[13] A. A. Hadi, A. W. Mohamed, and K. M. Jambi, “Lshade-spa memetic framework for solving large-scale optimization problems,” *Complex & Intelligent Systems*, Dec 2018. [Online]. Available: https://doi.org/10.1007/s40747-018-0086-8

[14] J. L. Morales and J. Nocedal, “Remark on algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound constrained optimization,” *ACM Trans. Math. Softw.*, vol. 38, no. 1, pp. 7:1–7:4, Dec. 2011. [Online]. Available: http://doi.acm.org/10.1145/2049662.2049669