SHADE with Iterative Local Search for Large-Scale Global Optimization

1st Daniel Molina
Department of Computer Science
University of Granada
Granada, Spain
dmolina@decsai.ugr.es

2nd Antonio LaTorre
DATSI, ETSIINF
Center for Computational Simulation
Universidad Politécnica de Madrid
Madrid, Spain
atorre@fi.upm.es

3rd Francisco Herrera
Department of Computer Science
University of Granada
Granada, Spain
herrera@decsai.ugr.es

***Abstract*—Global optimization is a very important topic in research due to its wide applications in many real-world problems in science and engineering. Among optimization problems, dimensionality is one of the most crucial issues that increases the difficulty of the optimization process. Thus, Large-Scale Global Optimization, optimization with a great number of variables, arises as a field that is getting an increasing interest. In this paper, we propose a new hybrid algorithm especially designed to tackle this type of optimization problems. The proposal combines, in a iterative way, a modern Differential Evolution algorithm with one local search method chosen from a set of different search methods. The selection of the local search method is dynamic and takes into account the improvement obtained by each of them in the previous intensification phase, to identify the most adequate in each case for the problem. Experiments are carried out using the CEC'2013 Large-Scale Global Optimization benchmark, and the proposal is compared with other state-of-the-art algorithms, showing that the synergy among the different components of our proposal leads to better and more robust results than more complex algorithms. In particular, it improves the results of the current winner of previous Large-Scale Global Optimization competitions, Multiple Offspring Sampling, MOS, obtaining very good results, especially in the most difficult problems.**

***Index Terms*—Large-scale Global Optimization, Differential Evolution, Memetic Computing, Hybridization.**

I. Introduction

Continuous optimization is an important research field because it appears in many real-world optimization problems in very different knowledge areas. In this kind of problems, the solution can be formulated as a vector of a certain length of continuous variables within a domain search. Evolutionary Algorithms [1] such as Differential Evolution, DE [2], [3], are very useful for solving this type of problems, because they can obtain accurate solutions in complex problems without specific information about them, something very important in many real-world problems [4]. However, they are very sensitive to the size of the problem, as the domain search increases exponentially with the number of dimensions.

Large-Scale Global Optimization, LSGO, is a particular category of global optimization in which the problem size reaches (or exceeds) one thousand of variables. In this type of optimization the efficiency of the search techniques is crucial, due to the huge domain search to explore. In recent years, many LSGO special sessions are been carried out with different algorithms specially designed for this type of optimization. One algorithmic design technique that is gaining a lot of attention is the partitioning of the problem into smaller ones. One example of this is the different grouping variable techniques developed by some authors [5], [6]. However, the current state-of-art and winner of LSGO competitions since 2013, Multiple Offspring Sampling, MOS [7], [8], follows a different approach: it dynamically combines different search techniques that are used simultaneously and their participation in the overall search process is adjusted according to their part performance.

In this paper we propose a new algorithm, SHADE with an iterative Local Search, SHADE-ILS, that combines the exploration power of a recent DE algorithm with the exploitation ability of several local search, LS, methods. At each iteration of the algorithm, the DE is applied to evolve the population of candidate solutions and a LS is used to improve the current best solution. The LS techniques is selected at each iteration according to the previous relative improvement obtained by the applications of each LS method.

This algorithm is based on the one proposed in [9], IHDELS, but with a number of important differences: First, the selection of the LS to apply at each iteration has been improved, as will be shown later. Second, a new restart mechanism that detects stagnation has been introduced. Finally, in this proposal a more powerful DE, SHADE [10] is applied instead of the previously DE algorithm, SaDE [11].

We have compared the proposed method with IHDELS, and also with other reference algorithms using the CEC'2013 benchmark for LSGO [12]. The results obtained show that the new algorithm improves previous proposals. Moreover, SHADE-ILS improves the results of the current winner of past LSGO competitions, Multiple Offspring Optimization, MOS [7], [8], which, as far as the authors are concerned, had not been improved since its proposal back in 2013, which gives and idea of how difficult this is.

This work has the following structure: In Section II, the SHADE-ILS algorithm is described in detail, highlighting the main differences with the previous algorithm. In Section III, we analyze the results obtained by our proposal compared with other reference state-of-the-art algorithms. Finally, in Section

IV main conclusions and future work are summarized.

II. Proposal

In this section we are going to describe the proposed algorithms, SHADE-ILS, highlighting the changes from the original IHDELS. For detailed information on this previous version of the algorithm, the author is referred to [9].

Algorithm 1 General algorithm SHADE-ILS

1: $population \leftarrow random(dim, popsize)$
2: $initial\_solution \leftarrow (upper + lower)/2$
3: $current\_best \leftarrow LS(initial\_solution)$
4: $best\_solution \leftarrow current\_best$
5: **while** $totalevals < maxevals$ **do**
6: $current\_best \leftarrow SHADE(population, current\_best)$
7: $previous \leftarrow current\_best.fitness$
8: $improvement \leftarrow previous - current\_best.fitness.$
9: *Chose the LS method to apply in this iteration.*
10: $current\_best \leftarrow LS(population, current\_best)$
11: *Update probability of applying LS for next iterations.*
12: **if** $better(current\_best, best\_solution)$ **then**
13: $current\_best \leftarrow best\_solution.$
14: **end if**
15: **if** *Must restart* **then**
16: *Restart and update* $current\_best.$
17: **end if**
18: **end while**

Algorithm 1 shows the general scheme of the algorithm. As can be observed, the algorithm applies, in an iterative way, DE and a LS method, exploring all the variables at the same time (an important difference compared to algorithms using variable grouping techniques). Another important characteristic is that the algorithm maintains the same population between DE calls. Additionally, the parameters of the LS method are also maintained when it is applied multiple times to the same solution (except when the algorithm is restarted). Thus, the workflow of the algorithm can be summarized in two steps:

- An exploratory technique is first applied to explore the search space. We choose the recent SHADE [10] algorithm because it is simple and self-adapts its parameters. There is a reducing-population version, L-SHADE [13] very popular in the field of continuous optimization. However, in our case, the population size adjustment would reduce too quickly the exploration.
- An intensification LS algorithm is chosen at each iteration from two different LS methods. One is the MTS LS-1 algorithm [14], especially designed for LSGO. The other one is the classic L-BFGS-B [15] that uses an approximation of the gradient to improve the search. These methods are complementary: MTS is very fast and appropriated for separable problems, but it is very sensitive to rotations. However, while L-BFGS-B is less powerful it is less sensitive to rotations.

This overall algorithmic framework described in Algorithm 1 is shared between this and the previous proposal [9]. However, as stated before, there are three main differences between these proposals: the DE used (SHADE instead of SaDE) (line 6), the selection of the LS method (lines 9, 11), and the restart mechanism (lines 15-17). In the following, we provide details of the main features of our new proposal.

A. Exploratory Algorithm: SHADE

In this work we apply SHADE as the exploratory component. This DE algorithm has the following advantages:

- It has a very advanced self-adaptation of the DE parameters, CR and F, allowing for a good adaptation to each problem. The only required parameter is the population size.
- The mutation operator takes into account previous solutions stored in an archive, increasing thus the diversity of the new solutions.
- The mutation operator is biased towards not always selecting the best solution. Instead, it randomly selects among the best p solutions.

A detailed description of the SHADE algorithm can be found in [10].

In recent new versions of SHADE, some authors consider it to be too exploratory and thus a lineal population reduction is usually applied [13], [16]. However, in our case, we are interested in the algorithm as the exploratory component because we have another intensification algorithm, so the population reduction is not needed to improve its performance. Furthermore, we have tested both SHADE and L-SHADE, obtaining the best results with SHADE.

B. LS Method selection

In IHDELS, the selection of the LS method at each step was carried out with a certain probability to chose each LS method (line 9) P_{LS}. This value is initialized as $P_{LS} = \frac{1}{|LS|}$, where $|LS|$ is the number of LS methods (2 in our case). Also, at each iteration (line 11) the improvement for each LS method is computed as:

$$I_{LS} = \frac{fitness(Before_{LS}) - fitness(After_{LS})}{Before_{LS}} \quad (1)$$

Then, the probability of choosing each LS was updated as a function of the average I_{LS} for each LS method.

In our new proposal the improvement of each LS method is obtained in the same way (Equation 1). However, instead of using the average I_{LS} for each LS algorithm, it selects the LS with largest I_{LS} in its last application. This criterion is both simpler and, in our experiments, more efficient than the previous one. Moreover, when a LS method quickly decreases its performance, the use of the average I_{LS} takes more time to detect this change, whereas using the previous I_{LS} provides a faster adaptation (a LS method is applied until the previous application is worse than the improvement of the other LS method).

C. Restart mechanism

In IHDELS, the restart criterion (line 16) was only met when no improvement was obtained during a full iteration. However, in real-parameter optimization it is common to keep slightly improving the best solution by very small factors, which is normally not enough. This made the restart mechanism to be actually applied in just a few cases, and in those cases, the restart did not actually improve the results.

In this work, we propose a new restart mechanism that is applied when, during three consecutive iterations, the ratio of improvement (considering both the DE and the LS application) is lower than 5%. In those cases, the restart mechanism is applied as follows:

- A solution sol from the population is randomly selected.
- A small disturbing is done to sol with a normal distribution with mean 0 and 10% over the domain range as standard deviation: $current_{best} = sol + rand_i \cdot 0.1 \cdot (b-a)$, where $rand_i$ returns a random number $rand_i \in [0,1]$ and $[a,b]$ is the domain search.
- The population of the DE algorithm is randomly restarted.
- The adaptive parameters of the LS methods are restarted to their default values.

Usually, algorithms restart again from a randomly solution (or a disturbed solution from the current best). We propose to choose randomly a solution of the population used by the DE. The idea is to chose a relative good solution in the population but still not improved by the LS method. Then, a small modification to this solution is carried out to avoid the population to be too similar.

Additionally, when the LS does not improve the current best at all, its self-adapted parameters are also restarted. Considering three full iterations as the criterion to restart the full population follows the rationale of allowing the algorithm to apply both local search methods and also to restart their parameters. If no significant improvements are obtained after that, the restart of the population as described previously takes place.

III. Experimentation

Experiments have been carried out using the benchmark and the experimental conditions used in the CEC2013 LSGO competition [12]. This benchmark is made up of 15 optimization functions with 1000 dimensions, and several degrees of separability, from completely separable functions to fully-non-separable ones.

- Fully separable functions: $f_1 - f_3$.
- Partially separable functions: with a separable subcomponent ($f_4 - f_7$) and without separable subcomponents ($f_8 - f_{11}$).
- Overlapping functions: $f_{12} - f_{14}$.
- Non-separable functions: f_{15}.

A report with detailed information on the benchmark can be found in [12].

Each algorithm is run for each function 51 times, and each run finishes when a maximum number of evaluations, *fitness evaluations, FEs*, is reached ($3 \cdot 10^6$ in this case). Additionally, the best *fitness* is measured at different milestones (in terms of FEs). In particular, the following milestones are considered for our proposal: $\{1.2, 3.0, 6.0, 9.0, 12, 15, 18, 21, 24, 27, 30\} \cdot 10^5$. The best fitness for each milestone is automatically recorded by the benchmark code. However, in previous competitions only a subset of the milestones was considered: $1.2 \cdot 10^5$, $6.0 \cdot 10^5$, $3.0 \cdot 10^6$, so we are going to use only these last milestones to allow a straightforward comparison with state-of-the-art algorithms.

The parameters values used in this experimentation are shown in Table I. It can be observed that each iteration uses 50000 evaluations (of which 25000 are for DE and the other 25000 for the LS method). The remaining parameters shared by IHDELS and the proposed SHADE-ILS are kept the same.

TABLE I: Parameters values

Algorithm	Parameter	Description	value
Shared Parameters	DE popsize	Population size	100
	FE_{DE}	FEs in DE application	25000
	FE_{LS}	FEs in LS application	25000
	MTS_{Istep}	Initial stepsize for MTS-LS1	20
IHDELS	Freq_{LS}	Frequency of prob. update	10
SHADE-ILS	$Restart_N$	Times without improvement	3
IHDELS	$Threshold$	Minimum ratio improvement	5%

The results of our proposal are summarized in Table VII, as requested by the organizers of the WCCI 2018 LSGO Competition.

In the following sections we discuss on these results. First, we are going to analyze the influence of the different components of our proposal, paying special attention to the restart mechanism. Then, we are going to compare our results against those of the previous version of the algorithm, IHDELS. Finally, we compare our proposal against the current unbeaten winner of LSGO Competitions, MOS. Due to page limits, our analysis will be limited to when the maximum number of FEs has been reached: $3.0 \cdot 10^6$ FEs.

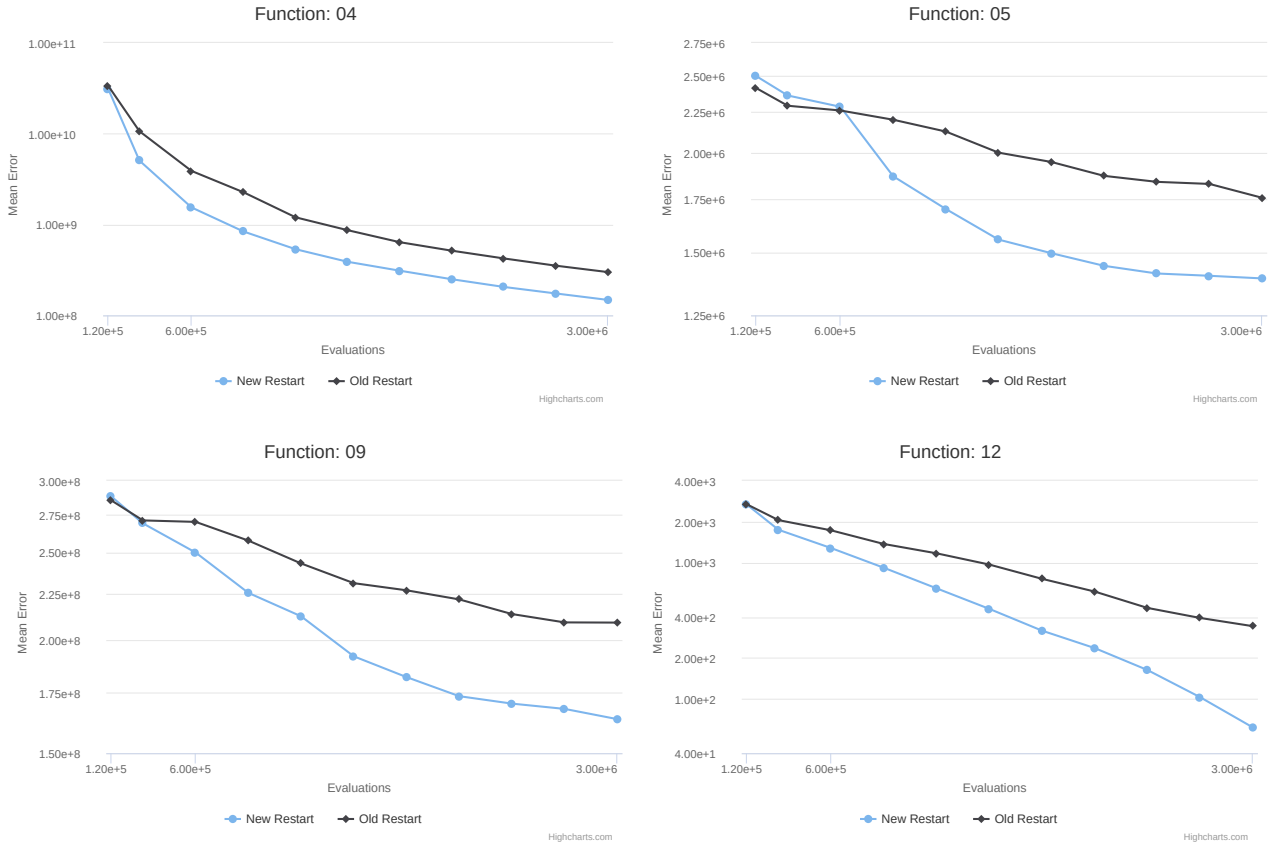
A. Influence of the different components

In this section, we analyze the contribution of each of the subcomponents of SHADE-ILS to the overall results.

First, we are going to study the contribution of the new restart mechanism and illustrate the main differences when compared with the previous method by taking into account several representative functions (Figure 1). There are some functions, such as F_4, in which the new restart mechanism improves the old one but keeping a similar convergence trend. In other cases, such as, for example, function F_5, the new restart technique allows the algorithm to fully explore a basin of attraction, which leads to very fast improvements of the best solution. Finally, we have observed, for several functions (such as F_9 or F_{12}) that the new restart mechanism is able to significantly increase the convergence speed of the algorithm.

Now, we are going to compare the influence of the different components individually. Table II shows the results obtained

Fig. 1: Average error using the new and the old restart mechanisms



by original IHDELS version and the different improvements referred in Section II (from right to left):

- Original IHDELS proposed in 2015 [9], that uses SaDE and the original restart mechanism.
- Algorithm exchanging SaDE by SHADE but keeping the same restart mechanism.
- Algorithm with the new restart mechanism but using the SaDE algorithm instead of SHADE.
- Final proposal, using both SHADE instead of SaDE and the new restart mechanism.

In the previous analysis we did not incorporate the selection method for the LS method because the differences were not as clear as for the other improvements. However, we decided to keep that change because it reduces the algorithmic complexity and removes one parameter ($Freq_{LS}$) without deteriorating the results.

By observing Table II we can make the following conclusions on the influence of each of the components:

- The change in the restart mechanism has a more important effect than that on the DE component.
- The use of SHADE instead of SaDE greatly reduces the error using any restart mechanism, especially in non-separable and overlapping functions.
- The combination using both SHADE and the new restart mechanism clearly beats any other alternative. Thus, it can be observed that the good results are obtained by these two components when used simultaneously.
- The proposed algorithm considering both improvements obtains the best results in 12 of the 15 functions, and in

TABLE II: Results for the $3 \cdot 10^6$ FEs milestones and the different combinations of the components of the algorithm

Func.	Using SHADE +New Restart	Using SaDE +New Restart	Using SHADE +Old Restart	IHDELS
F_1	2.69e-24	1.21e-24	1.76e-28	**4.80e-29**
F_2	**1.00e+03**	1.26e+03	1.40e+03	1.27e+03
F_3	2.01e+01	2.01e+01	2.01e+01	**2.00e+01**
F_4	**1.48e+08**	1.58e+08	2.99e+08	3.09e+08
F_5	**1.39e+06**	3.07e+06	1.76e+06	9.68e+06
F_6	**1.02e+06**	1.03e+06	1.03e+06	1.03e+06
F_7	**7.41e+01**	8.35e+01	2.44e+02	3.18e+04
F_8	**3.17e+11**	3.59e+11	8.55e+11	1.36e+12
F_9	**1.64e+08**	2.48e+08	2.09e+08	7.12e+08
F_{10}	**9.18e+07**	9.19e+07	9.25e+07	9.19e+07
F_{11}	5.11e+05	**4.76e+05**	5.20e+05	9.87e+06
F_{12}	**6.18e+01**	1.10e+02	3.42e+02	5.16e+02
F_{13}	**1.00e+05**	1.34e+05	9.61e+05	4.02e+06
F_{14}	**5.76e+06**	6.14e+06	7.40e+06	1.48e+07
F_{15}	**6.25e+05**	8.69e+05	1.01e+06	3.13e+06
Better	12	1	0	2

the other functions the differences are very small.

To summarize, the different components of SHADE-ILS, especially the new restart mechanism and the use of SHADE instead of SaDE, are able to improve the results of previous IHDELS. Furthermore, the combination of all these components obtains the best overall results, which are significantly better than any of the other combinations on their own.

B. Comparing SHADE-ILS against IHDELS

In Table II it can be observed that SHADE-ILS is better than IHDELS in 13 of the 15 functions (with minimum difference in the other two). Besides, the improvements of SHADE-ILS are especially important in the most difficult functions, obtaining an error at least one order of magnitude lower in many of them (see, for example, functions $F_5, F_7, F_8, F_{11}, F_{12}, F_{13}, F_{14},$ or F_{15}). Also, by changing the method of selection of the LS algorithm, SHADE-ILS has one less parameter, Feq_{LS}, while keeping a more robust behavior. However, in Table III we can observe that the SHADE-ILS is in average a 15% slower (both were implemented in Python).

TABLE III: Time required for each algorithm and function

Function	SHADE-ILS	IHDELS	Difference (in %)
F_1	13m30s	18m50s	-29
F_2	26m09s	22m44s	13
F_3	25m07s	22m38s	9
F_4	22m23s	19m34s	12
F_5	26m28s	23m31s	11
F_6	26m30s	21m32s	18
F_7	11m58s	8m01s	33
F_8	27m18s	23m06s	15
F_9	32m03s	27m16s	14
F_{10}	31m20s	24m47s	20
F_{11}	25m20s	20m43s	18
F_{12}	6m21s	2m23s	62
F_{13}	25m04s	20m43s	17
F_{14}	25m13s	20m34s	18
F_{15}	19m43s	15m33s	21

C. Comparing SHADE-ILS against MOS

In this section we are going to compare our proposal against MOS, the state-of-the-art algorithm in LSGO and, since 2013, the winner of all the LSGO competitions. No other algorithm in these LSGO competitions has been able to improve its results, so it is the clear reference algorithm to compare to.

In Tables IV-VI we provide the results of SHADE-ILS and MOS for the different milestones ($1.2 \cdot 10^5$, $6 \cdot 10^5$, and $3 \cdot 10^6$, respectively), obtaining the following conclusions:

- While MOS obtains better results at $1.2 \cdot 10^5$ FEs, both algorithms are very similar at $6 \cdot 10^5$ FEs, and for the maximum number of FEs, $3 \cdot 10^6$, SHADE-ILS gets the best results in 10 of the 15 functions.
- While MOS continues to be better in separable functions (f_1-f_3) SHADE-ILS is better for more complex ones: with the exception of functions f_6, f_{10}, SHADE-ILS is clearly better in all the other functions.

TABLE IV: Results obtained by SHADE-ILS against MOS for FEs=$1.2 \cdot 10^5$

Functions	IHSHADELS	MOS
F_1	**6.10e+04**	2.71e+07
F_2	2.65e+03	**2.64e+03**
F_3	2.03e+01	**7.85e+00**
F_4	**3.13e+10**	3.47e+10
F_5	**2.50e+06**	6.96e+06
F_6	1.05e+06	**3.11e+05**
F_7	3.95e+08	**3.46e+08**
F_8	**2.12e+14**	3.72e+14
F_9	**2.88e+08**	4.29e+08
F_{10}	9.43e+07	**1.16e+06**
F_{11}	6.55e+09	**3.13e+09**
F_{12}	**2.67e+03**	1.16e+04
F_{13}	1.29e+10	**8.37e+09**
F_{14}	1.62e+11	**4.61e+10**
F_{15}	9.12e+07	**1.45e+07**

TABLE V: Results obtained by SHADE-ILS against MOS for FEs=$6 \cdot 10^5$

Functions	SHADE-ILS	MOS
F_1	**3.71e-23**	3.48e+00
F_2	1.80e+03	**1.78e+03**
F_3	2.01e+01	**1.33e-10**
F_4	**1.54e+09**	2.56e+09
F_5	**2.29e+06**	6.95e+06
F_6	1.04e+06	**1.48e+05**
F_7	**9.25e+05**	8.19e+06
F_8	**6.93e+12**	8.41e+13
F_9	**2.50e+08**	3.84e+08
F_{10}	9.29e+07	**9.03e+05**
F_{11}	**1.37e+08**	8.05e+08
F_{12}	**1.28e+03**	2.20e+03
F_{13}	**5.68e+07**	8.10e+08
F_{14}	**6.97e+07**	2.03e+08
F_{15}	1.22e+07	**6.26e+06**

- SHADE-ILS is very competitive in more functions, especially in overlapping and non-separable ones. Not only it improves MOS in many cases, but also results are at least one order of magnitude lower in many cases.

D. Overall comparison

To conclude this study, we are going to apply the LSGO competition comparative procedure. In recent LSGO competitions, since CEC'2013, all the algorithms are assigned points using the following process:

1) For each function, algorithms are sorted by their average mean error.
2) Each algorithm is given a certain number of points, according to its ranking, and following the F-1 criterion: 25 points to the best algorithm, 18 points to the runner-up, 15 to the third one, etc.
3) The results for each function aggregated for all the algorithms to obtain their overall score.

TABLE VI: Results obtained by SHADE-ILS against MOS for FEs=$3 \cdot 10^6$

Functions	SHADE-ILS	MOS
F_1	2.69e-24	**0.00e+00**
F_2	1.00e+03	**8.32e+02**
F_3	2.01e+01	**9.17e-13**
F_4	**1.48e+08**	1.74e+08
F_5	**1.39e+06**	6.94e+06
F_6	1.02e+06	**1.48e+05**
F_7	**7.41e+01**	1.62e+04
F_8	**3.17e+11**	8.00e+12
F_9	**1.64e+08**	3.83e+08
F_{10}	9.18e+07	**9.02e+05**
F_{11}	**5.11e+05**	5.22e+07
F_{12}	**6.18e+01**	2.47e+02
F_{13}	**1.00e+05**	3.40e+06
F_{14}	**5.76e+06**	2.56e+07
F_{15}	**6.25e+05**	2.35e+06

4) Algorithms are compared (for example, with bar plots, or stacked bar plots) for each considered milestone.

According to the previous procedure, we have conducted an overall comparison considering the following algorithms: Original IHDELS [9], MOS algorithm [7], and our proposal, SHADE-ILS.

The results of this comparative are shown in Figure 2. In this figure, it can be observed that SHADE-ILS obtains the best results for FEs = $6 \cdot 10^5$ and $3 \cdot 10^6$, obtaining the highest number of points in most of the categories.

IV. Conclusions

In this paper we have proposed a new optimization algorithm especially designed for Large Scale Global Optimization, SHADE-ILS, that combines the global exploration ability of an adaptive DE algorithm with two intensification LS methods to continuously improve the results. In each iteration, it evolves the population with SHADE and then it chooses the LS with the best relative improvement during the last activation phase to improve the current best solution found by SHADE. A restart mechanism has been incorporated to the algorithm to allow the exploration of new regions of the search space when the search gets stagnated (i.e., when the relative improvement is low during several consecutive iterations). The restart selects one solution for slightly modification before applying the aforementioned local search methods to it.

In the experimental section we have tested and analyzed SHADE-ILS using the benchmark proposed for the CEC'2013 competition on LSGO. First, we have compared the contribution of each modification on top of the original IHDELS (selection of SHADE vs SaDE, restart mechanism) to improve the results, concluding that they significantly contribute to improve the results. To continue, we have compared our proposal with several reference algorithms for LSGO, including the current winner in previous competitions, MOS. In this comparison, SHADE-ILS obtains the best overall results, beating MOS for the first time since CEC'2013. This comparison reveals that SHADE-ILS is especially good for the most complex functions, with overlapping and non-separable components, achieving not only the minimum error for most of them, but also an error one order of magnitude lower than the previous one.

As a future work, we are going to try new LS methods to accelerate the convergence rate of the algorithm and to obtain better results in more non-separable functions.

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 Regional Government (P12-TIC-2958).

References

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

[2] K. V. Price, R. M. Rainer, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Springer-Verlag, 2005.

[3] S. Dasa, S. Maity, B.-Y. Qu, and P. Suganthan, "Real-parameter evolutionary multimodal optimization-a survey of the state-of-the-art," *Swarm and Evolutionary Computation*, vol. 1, no. 2, pp. 71–78, 2011.

[4] N. Xiong, D. Molina, M. L. Ortiz, and F. Herrera, "A walk into metaheuristics for engineering optimization: principles, methods and recent trends," *International Journal of Computational Intelligence Systems, 2015*, vol. 8, no. 4, pp. 606–636, June 2015.

[5] M. N. Omidvar, X. Li, and X. Yao, "Cooperative Co-evolution with delta grouping for large scale non-separable function optimization," in *2010 IEEE Congress on Evolutionary Computation (CEC 2010)*. IEEE, 2010, pp. 1762–1769.

[6] 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.

[7] A. LaTorre, S. Muelas, and J. M. Peña, "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.

[8] A. LaTorre, S. Muelas, and J. Peña, "A Comprehensive Comparison of Large Scale Global Optimizers," *Information Sciences*, vol. 316, pp. 517–549, 2014.

[9] D. Molina and F. Herrera, "Iterative hybridization of DE with local search for the CEC'2015 special session on large scale global optimization," in *2015 IEEE Congress on Evolutionary Computation (CEC 2015)*. IEEE, 2015, pp. 1974–1978.

[10] R. Tanabe and A. Fukunaga, "Evaluating the performance of shade on cec 2013 benchmark problems," in *2013 IEEE Congress on Evolutionary Computation*, June 2013, pp. 1952–1959.

[11] A. Qin, V. Huang, and P. Suganthan, "Differential Evolution Algorithm With Strategy Adaptation for Global Numerical Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, Apr. 2009.

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

[13] R. Poláková, J. Tvrdík, and P. Bujok, "L-shade with competing strategies applied to cec2015 learning-based test suite," in *2016 IEEE Congress on Evolutionary Computation (CEC)*, 2016, pp. 4790–4796.

[14] L. Y. Tseng and C. Chen, "Multiple Trajectory Search for Large Scale Global Optimization," in *2008 IEEE Congress on Evolutionary Computation (CEC 2008)*. IEEE Press, Jun. 2008, pp. 3052–3059.

[15] 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.

[16] J.Brest and B. M.Sepesy Mačec, "iL-SHADE: Improved L-SHADE Algorithm for Single Objective Real-Parameter OptimizationTesting United Multi-operator Evolutionary Algorithms-II on Single Objective," in *2016 IEEE Congress on Evolutionary Computation (CEC)*, 2016, pp. 1188–1195.

Fig. 2: Comparison using the CEC'2013 benchmark criterion

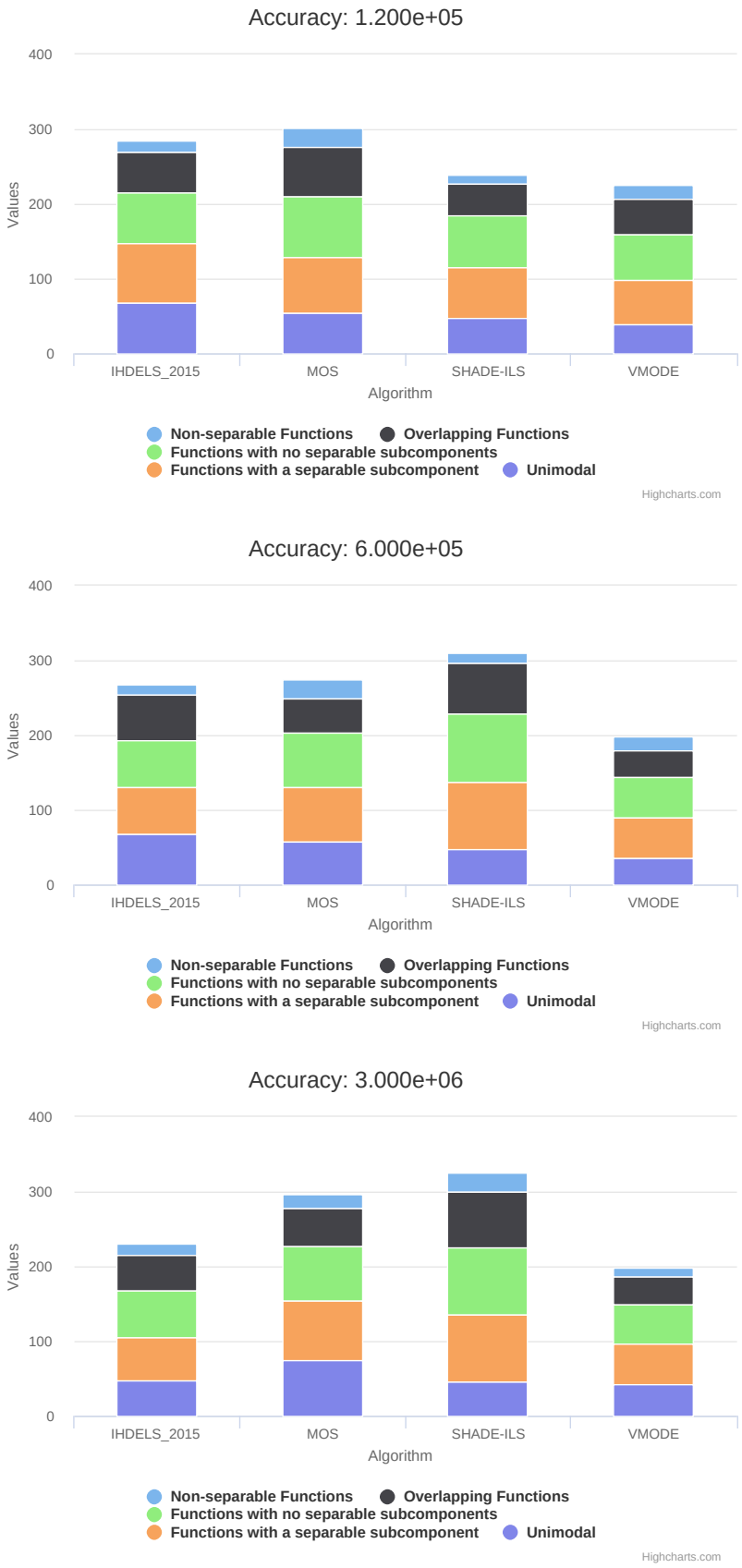


TABLE VII: Results for SHADE-ILS in the LSGO competition

Milestone	Category	f_1	f_2	f_3	f_4
	Best	1.554524e+04	2.245803e+03	2.008179e+01	1.534228e+10
	Median	6.071679e+04	2.565400e+03	2.011079e+01	2.598669e+10
1.20e+05	Worst	1.415877e+05	3.070720e+03	2.116410e+01	7.496997e+10
	Mean	6.095098e+04	2.652277e+03	2.027693e+01	3.128835e+10
	Std	3.034470e+04	2.172362e+02	3.922866e-01	1.425941e+10
	Best	0.000000e+00	1.425385e+03	2.002992e+01	7.601136e+08
	Median	5.073051e-25	1.736995e+03	2.007101e+01	1.258685e+09
6.00e+05	Worst	2.271773e-22	2.443857e+03	2.010646e+01	4.259609e+09
	Mean	3.711293e-23	1.798330e+03	2.007314e+01	1.543243e+09
	Std	6.423024e-23	2.927754e+02	2.242370e-02	8.217022e+08
	Best	0.000000e+00	8.528490e+02	2.002992e+01	5.946007e+07
	Median	0.000000e+00	9.884556e+02	2.005106e+01	1.151292e+08
3.00e+06	Worst	6.733050e-23	1.206723e+03	2.007981e+01	3.855623e+08
	Mean	2.693810e-24	9.999068e+02	2.005302e+01	1.478005e+08
	Std	1.346598e-23	8.895892e+01	1.115749e-02	8.720936e+07

Milestone	Category	f_5	f_6	f_7	f_8
	Best	1.529992e+06	1.037335e+06	1.746284e+08	3.219394e+13
	Median	2.504248e+06	1.048820e+06	3.611175e+08	1.617712e+14
1.20e+05	Worst	3.276053e+06	1.062440e+06	9.456264e+08	1.061200e+15
	Mean	2.498303e+06	1.050208e+06	3.948877e+08	2.120319e+14
	Std	3.851667e+05	6.675342e+03	1.537156e+08	2.038719e+14
	Best	1.294038e+06	1.016493e+06	4.721704e+05	8.823931e+11
	Median	2.284612e+06	1.039437e+06	7.657899e+05	6.174521e+12
6.00e+05	Worst	3.081915e+06	1.050850e+06	2.703706e+06	1.943808e+13
	Mean	2.285643e+06	1.037430e+06	9.250783e+05	6.926913e+12
	Std	3.669225e+05	8.346071e+03	4.728707e+05	4.277637e+12
	Best	1.092659e+06	1.002988e+06	7.645817e+00	1.814530e+10
	Median	1.412629e+06	1.024128e+06	5.463577e+01	2.784808e+11
3.00e+06	Worst	1.863953e+06	1.041058e+06	1.990906e+02	1.325738e+12
	Mean	1.391026e+06	1.023325e+06	7.405165e+01	3.172915e+11
	Std	2.030262e+05	1.188233e+04	5.456678e+01	3.061763e+11

Milestone	Category	f_9	f_{10}	f_{11}	f_{12}
	Best	2.153293e+08	9.317447e+07	1.982077e+09	2.093383e+03
	Median	2.884172e+08	9.432947e+07	4.230243e+09	2.574926e+03
1.20e+05	Worst	3.518991e+08	9.499023e+07	3.463071e+10	4.076292e+03
	Mean	2.881798e+08	9.427664e+07	6.550365e+09	2.673992e+03
	Std	3.484803e+07	4.899115e+05	7.258259e+09	4.864545e+02
	Best	1.930272e+08	9.075412e+07	7.850823e+07	5.974687e+02
	Median	2.511487e+08	9.300100e+07	1.240306e+08	1.327719e+03
6.00e+05	Worst	3.068166e+08	9.388529e+07	2.728405e+08	1.885590e+03
	Mean	2.498807e+08	9.288852e+07	1.371227e+08	1.280923e+03
	Std	3.082944e+07	7.639143e+05	4.737705e+07	2.981148e+02
	Best	1.298556e+08	9.058256e+07	3.102985e+05	8.531649e-20
	Median	1.629812e+08	9.197457e+07	4.499355e+05	3.986624e+00
3.00e+06	Worst	1.939016e+08	9.309367e+07	1.426263e+06	2.986941e+02
	Mean	1.635679e+08	9.181196e+07	5.105254e+05	6.182378e+01
	Std	1.567818e+07	6.930247e+05	2.248730e+05	1.039156e+02

Milestone	Category	f_{13}	f_{14}	f_{15}
	Best	4.740663e+09	5.152026e+10	6.119018e+07
	Median	1.302811e+10	1.138420e+11	8.108226e+07
1.20e+05	Worst	2.566986e+10	5.119414e+11	1.576511e+08
	Mean	1.287682e+10	1.618907e+11	9.124505e+07
	Std	5.970486e+09	1.082795e+11	2.566399e+07
	Best	1.721130e+07	4.623974e+07	5.755326e+06
	Median	4.100592e+07	6.416073e+07	1.286620e+07
6.00e+05	Worst	3.841661e+08	1.244856e+08	3.101329e+07
	Mean	5.678643e+07	6.967345e+07	1.217493e+07
	Std	7.269764e+07	1.824442e+07	6.300782e+06
	Best	3.516991e+04	4.925270e+06	3.910423e+05
	Median	8.276128e+04	5.745030e+06	6.090023e+05
3.00e+06	Worst	3.313440e+05	6.390782e+06	1.560353e+06
	Mean	1.002738e+05	5.760313e+06	6.254099e+05
	Std	7.187405e+04	3.757684e+05	2.402833e+05