Applying Memetic algorithm with Improved L-SHADE and Local Search Pool for the 100-digit challenge on Single Objective Numerical Optimization

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

2nd Francisco Herrera
DASCI Andalusian Institute of Data Science and Computational Intelligence, University of Granada
Granada, Spain
herrera@decsai.ugr.es

***Abstract*—In this paper, we have proposed a new optimization algorithm, Memetic improved L-SHADE with a local search pool, MiLSHADE-LSP, a memetic algorithm that combines an improved L-SHADE with a local search pool. Improved L-SHADE modifies several important parameters during the run to encourage exploration in initial stages and to focus later the search around the most promising solutions. The local search pool is responsible to continuously improve the best solutions. MiLSHADE-LSP uses a pool of two different local search, LS, methods, the Broyden-Fletcher-Goldfarb-Shanno method with limited memory, L-BFGS-B, and the Solis-Wets algorithms, with an adaptive mechanism to choose which one of them is applied in each iteration selecting which had obtained a greater improvement last time it was applied. In order to avoid waste LS applications, the proposed algorithm stores a list of individuals that were not previously improved by each LS method. It also includes a restart mechanism to explore new areas when the search is stuck, restarting the population but maintaining the best found solution, and resetting the LS Pool parameters. In the experimental section we have tested and analyzed MiLSHADE-LSP using the proposed benchmark for the competition *100-digit challenge on Single Objective Numerical Optimization*, obtaining that the LS Pool improves the algorithm, both achieving more optima and with a better performance. Results obtained show that MiLSHADE-LSP is a very competitive algorithm.**

***Index Terms*—Continuous optimization, global optimization, memetic algorithm, single objective numerical optimization, numerical optimization, differential evolution.**

I. Introduction

There are many optimization problems in which there is not any suitable and exact technique to solve it in a reasonable time. In these problems, Evolutionary Algorithms, EA [1], have arisen as a very good alternative, because they are able to obtain acceptable results with limited resources.

In order to obtain an adequate search, it is important to achieve a good trade-off between a exploration of all domain search and a exploitation of the most promising solutions. For doing that, hybrid meta-heuristics, like Memetic Algorithms, MA [2], [3], which combine different algorithms as components, are widely used. Initially, MAs were the combination of a population-based algorithm with a local search improvement, but nowadays there are more complex combination of different algorithms. Also, the convenience of using a pool of different local search methods has been proved [4].

Another important factor that can contribute to a good balance is focused on exploring in the early stages, and later, as the search progresses, on seeking more in depth around the best solutions. One option is to use initially a greater population to maintain a higher diversity at initial stages, and during the run to decrease the population size, as L-SHADE [5] does. Another strategy is to adapt several parameters during the search to enforce exploitation as the search evolves, as the improved L-SHADE [6].

In this work, we propose a new memetic algorithm, memetic improved L-SHADE with a local search Pool, MiLSHADE-LSP, which combines the continuous exploitation by using a local search pool with an EA, improved L-SHADE, iL-SHADE, that enforces the exploitation during the search. The main features of our proposal are:

- A decreasing population size, to enforce diversity at the beginning of the search, and focusing the search around most promising solutions. If the population prematurely converges, it is restarted.
- In each iteration, we are going to improve a promising solution by one local search, LS, method.
- The behavior of LS method strongly depends on the problem, so the proposal has a pool of LS methods, and it chooses each time which LS method is going to be applied.
- It stores when a solution was not previously improved by a LS method, to avoid applying it again, wasting resources.

We are going to test our proposal with the 100-digit challenge on single objective numerical optimization [7]. One interesting feature of this benchmark is that integrates difficult problems, and they are evaluated both the number of right digits achieved and the number of evaluations required to obtain them. Thus, although it could be considered a benchmark focused in the accuracy, the convergence speed of the algorithm is also taken in account. Results obtained show that proposed algorithm is very competitive.

This paper has the following structure. In Section II, we are going to describe the proposal in detail. In Section III, the experimental environment is introduced, showing the results obtained by the proposal, and analyzing the results. Finally, in Section IV, the main conclusions and several future works are summarized.

II. Proposal: Memetic with improved L-SHADE

In this paper, we propose a memetic algorithm with improved L-SHADE and a LS pool, MiLSHADE-LSP. The main features of this proposal are:

- The use of an advanced EA, the improved L-SHADE [6], responsible of the global exploration. This EA enforces global exploration at the beginning, focusing exploration around the most promising solutions during the run.
- A restart mechanism when the population prematurely converges.
- A LS Pool that contains different and complementary LS methods. The criterion used is to chose each time the algorithm whose last application had obtained a better relative improvement.
- A memory keeping, after the LS application, the solutions which were previous selected without a significant improvement. This memory is used to avoid applying a LS method many times without improvement.

In the following, we are going to describe the global scheme of the proposal, and later its components, the evolutionary algorithm responsible of the global exploration, the local search mechanism, and the restart mechanism.

A. Memetic framework

Algorithm 1 shows the global scheme of the proposal. It can be seen that in each iteration the population evolves as expected by the iLSHADE (lines 1-8).

When a new population is generated, one LS method from the LS Pool is selected, and the solution to apply it is chosen from the p best solutions. If there is no solution available, no LS is applied. The relative improvement is calculated, and it will be used for future selections (lines 17-24).

After the LS application, it is decided if the algorithm must be restarted, in that case the population is randomly reset, maintaining the current best solution in the next population (lines 25-27).

Algorithm 1 MiLSHADE-LSP scheme

```
1: $popsize \leftarrow popsize_{Initial}$.
2: $p \leftarrow p_{Initial}$.
3: $population \leftarrow random(dim, popsize)$.
4: $pop\_fitness \leftarrow fitness\_function(popsize)$.
5: $current\_best = min(pop\_fitness)$.
6: $current\_best \leftarrow LS(initial\_solution)$.
7: while $totalevals < maxevals$ do
8:     $population \leftarrow iLSHADE\_iteration(population, p)$.
9:     $totalevals \leftarrow totalevals + popsize$.
10:    $pop\_fitness \leftarrow fitness(popsize)$.
11:    $current\_best = min(pop\_fitness)$.
12:    Update parameter p.
13:    Update population size.
14:    if new popsize < previous popsize then
15:        population ← best individuals from population.
16:    end if
17:    Chose LS method from LS pool to apply.
18:    $solution_{LS}$ ← One of p best solutions not local optima.
19:    $previous \leftarrow fitness(solution_{LS})$.
20:    $solution_{LS}, evals_{LS} \leftarrow LS(solution, bounds)$.
21:    $totalevals \leftarrow totalevals + evals_{LS}$.
22:    $new\_fit \leftarrow fitness(solution)$.
23:    $improvement \leftarrow (previous - new\_fit)/previous$
24:    Update improvement history of applying $LS$.
25:    if Must restart then
26:        Restart maintaining current best individual.
27:    end if
28: end while
```

B. Improved L-SHADE

The exploratory algorithm is the improved L-SHADE [6] without any change. This algorithm has the following main features:

- A very advanced self-adaptation of the DE parameters, CR and F, allowing a good adaptation to each problem. The only required parameters are related to the population size.
- The mutation operator uses previous solutions stored in an archive, increasing 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.

In order to enforce more exploration at initial stages, and increasing selective pressure later, different parameters change during the run, according to the ratio of evaluations $r = numevals/maxevals$:

- The population size is linearly reduced during the run: $popsize = popsize_{Initial} - r \cdot (popsize_{Final} - popsize_{Initial})$.
  When the population is reduced, only the individuals with better fitness are maintained. This reduction produces a greater diversity in the initial population, to enforce exploration, leaving later in the population only the most promising solutions to focus the search around them.
- The parameter p used to indicate the number of best solutions considered for the mutation also is reduced: $p = p_{Initial} + r \cdot (p_{Final} - p_{Initial})$.
  Thus, in initial stages, more promising solutions are used to guide the search, and later only a few of them are used,

the best ones, to enforce selective pressure.

- The allowed values for CR and F depend on the number of iterations, by the following constraints:

$$\text{CR, F constraints} \begin{cases} CR \geq 0.5, F \leq 0.7 & \text{if } r \leq 0.25 \\ CR \geq 0.25, F \leq 0.8 & \text{if } r \leq 0.5 \end{cases}$$

  Thus, in initial stages, at least the half of variables for solutions are mutated, and later, a smaller ratio of variables can be mutated.

For more details, you can consult [6].

C. Local Search Application

The proposal, for each iteration, after the steps of mutation, crossover, selection, and population reduction (see previous subsection), applies a LS method to one solution, to try to improve even more its fitness (lines 17-23 in the Algorithm 1).

This process is done using a LS pool composed by different LS methods and feedback information about previous applications. In this proposal, the LS methods used are the well-known Solis-Wets method [8], that applies small changes using a normal distribution with an adaptive variance, and the classic L-BFGS-B [9] that uses an approximation of the gradient to improve the search. However, the LS methods could be changed without problem, the LS Pool is flexible enough to change both the LS methods considered and its number.

In the following, we indicate the steps of the LS phase:

a) Selection of the LS method: Initially, every one of the available LS methods are selected.

After each LS application, it is stored the last $Improvement_{LS}$, obtained by Equation 1:

$$Improvement_{LS} = \frac{previous\_fitness - new\_fitness}{previous\_fitness} \quad (1)$$

In each iteration, it is selected the LS method with a greater last $Improvement_{LS}$. It is a simple and efficient mechanism, capable of obtaining very good results [10].

b) Selection of the Solution to improve: It is selected one of the p (parameter already used by iL-SHADE) best solutions in the population. The idea is to select one solution that was not previously applied by the same LS method obtaining an insufficient improvement (using a threshold value, $Threshold_{LS}$). For doing that, for each LS Method it is kept a list of solutions which were previous selected without obtained enough improvement in the LS application.

First, the solution with best fitness is selected, only if it was improved enough. If not, it is selected randomly another one of the p best solutions, while it was not improved enough. It could happen that the LS method was previously applied to all p best solutions without significant improvement, in that case, the process finishes here.

c) Applying the LS method to the selected individual: In that step, if a solution was chosen, the selected LS method is applied to that individual during LS_{Eval} evaluations. For Solis-Wets' algorithm, an initial step size of I_{step} is applied.

d) Calculating the improvement and updating the decision parameters: After each LS method, it is calculated $Improvement_{LS}$ obtained by that LS following Equation 1. $Improvement_{LS}$ is used to update the previous value for the executed LS method.

If the relative improvement is lower than a $Threshold_{LS}$ value, solutions before and after the LS method are kept into the LS pool to avoid selecting them again (for the same LS method).

D. Restart technique

The final steps in each iteration decide if the population should be restarted (lines 25-27 in Algorithm 1).

- It is considered that the population must be restarted when the difference in fitness between the best and the worst solution in the population is lower than a $Threshold_{Restart}$ value.
- The restart implies that, at exception of the current best solution, all the other solutions in the populations are randomly generated, and their fitness calculated again. The restart takes in account the current population size (see subsection II-B).
- Additionally, not only the population is restarted, the LS_{Pool} parameters and memories are also updated (its history of last improvements, and the memory of solutions are also removed).

III. Experimental Section

A. 100-digit benchmark

The algorithm is tested using the benchmark, proposed by the 100-digit competition[1] [7].

This benchmark is composed by 10 different functions, and the goal is to achieve the optimum value (1.0 in all functions) with at least 10 decimal places. The 10 decimal places for the 10 problems are which give it the name 100-digit challenge.

The source code of the benchmark is available in C++ and Matlab [2], that we have used from Python with our own package, freely available by us including source code at https://github.com/dmolina/cec2019comp100digit.

The functions have several search domains (although the majority have range [-100, 100]), and different dimension (9 for f_1, 16 for f_2, 18 for f_3, and 10 for the others).

Experiments have been carried out with the following process:

- For each function, the algorithm is run 50 consecutive trials, each one with a different initial population randomly generated.
- It is counted the total number of correct digits of the 25 trials that give the best results.
- The number of evaluations for each level of digit accuracy is registered.

[1]http://cec2019.org/programs/competitions.html\#cec-06

[2]http://www.ntu.edu.sg/home/epnsugan/index_files/CEC2019/CEC2019.htm

- The score for that function is the average number of correct digits in the best 25 trials, and the total score is obtained.

At different of the majority of benchmarks, none stopping criterion is given, it is up to the researcher to decide when the algorithm should stop. That flexibility may suggest that MiLSHADE-LSP, that uses the ratio of evaluations inside its strategy, could not be a good option when an arbitrary $maxevals$ is used. However, as we will see later, it is not true, because we obtain very good results.

In the comparisons carried out, we analyze both the final scores (total and by functions) and the required number of evaluations to reach the number of decimal digits.

You can consult [7] to get more details about the benchmark and the experimental conditions.

B. Tuning of Parameters

Although it is acceptable to tune 2 parameter(s) independently for each problem, in MiLSHADE-LSP we are going to analyze results with and without the tuning. This double analysis is carried out because we want also to observe the behavior of our proposal considering each problem as a black-box one. The only information used is the dimension value (to define the initial parameter values), which is a parameter known for every real-world problem, and the ratio of evaluations (used to adapt both the population size and the p parameter used in the mutation method).

Component	Parameter	Value
iL-SHADE	Initial popsize	$12 \cdot dim$
	Final popsize	4
	p	from 0.2 to 0.1
LS_{Pool}	LS Methods	Solis Wets, L-BFGS-B
	LS_eval	100
	$Threshold_{LS}$	10^{-10}
Restart	$Threshold_{Restart}$	10^{-8}

TABLE I: Fixed parameter values

Component	Parameter	Default value
LS_{Pool}	I_{Step}	10%
iL-SHADE	$maxevals$	$100000 \cdot dim$

TABLE II: Default values for parameters used for tune

The proposal has the fixed parameter values indicated in Table I. The majority of parameters values used are the recommended values by the authors of iLSHADE [6], the only additional parameters are *LS_eval* and *I_Step*. In Table II, there are the default parameter values for parameters used in the tuned version. The maximum evaluation number could be consider high, but the majority of optima are obtained in a small percentage of the evaluations.

C. Results

In this section, we analyze the results obtained by MiLSHADE-LSP. First, we are going to observe if the changes made to iL-SHADE (mainly the introduction of the LS improvement) improve the results. Later, we are going to analyze the influence of the $maxevals$ parameter. Finally, we are going to analyze the results obtained with and without tuned parameters.

a) Improvement from non-memetic algorithm (iL-SHADE): In order to study the convenience of the memetic version of the iL-SHADE, first we are going to show the results, both of original iL-SHADE and the memetic version proposed in this work. The values of all shared parameters are the same.

Tables VIII and IX show the obtained results, as it was asked for the competition, for original iL-SHADE and MiLSHADE-LSP, respectively. These tables show, from the best 25 of the 50 runs, the number of times that each number of correct digits is achieved.

iL-SHADE	MiLSHADE-LSP
42.04	53.36

TABLE III: Score in original iL-SHADE vs non-tuned MiLSHADE-LSP ($maxevals = 100000 \cdot dim$)

We summarize the score of iL-SHADE and MiLSHADE-LSP in Table III. It can be observed that MiLSHADE-LSP score is higher than 53 (more than 10 points more than without LS). In details (results are in Table VIII for iL-SHADE and in Table IX for MiLSHADE-LSP), we can observe that both achieve the total of 10 correct digit in functions f_1, f_2, f_3, and f_5. However, using the LS method, the algorithm achieves more correct digits in more functions, f_4, f_6, f_8, and f_{10}, obtaining a better score. Other improvement of MiLSHADE-LSP is the reduction of evaluation number to achieve similar results. Table IV shows the reduction in number of evaluations to achieve the total digit reduction in functions optimized by both: f_1, f_2, f_3, and f_5. While in function f_2 the memetic version takes more evaluations, for the others functions the introduction of the LS greatly reduces the number of evaluations. In these problems, the optima are obtained with nearly a 18% of original number of evaluations. This is a very significant improvement in performance, specially in real-world problems.

Function	iLSHADE	MiLSHADE-LSP	Ratio
f_1	2894	307	10%
f_2	173041	287444	166%
f_3	1757291	72308	4%
f_5	68928	9819	14%
Total	2002154	369878	18%

TABLE IV: Ratio of evaluations iL-SHADE *vs* MiLSHADE-LSP in optimized functions

b) Influence of the maxevals parameter: MiLSHADE-LSP has a series of parameters whose values strongly depend on the current ratio of evaluations done. Thus, in our proposal, changing $maxevals$ (the maximum evaluation number) not

Function/maxevals	$10 \cdot 10^4 \cdot dim$	$25 \cdot 10^4 \cdot dim$	$50 \cdot 10^5 \cdot dim$
1	10.00	10.00	10.00
2	10.00	10.00	10.00
3	10.00	10.00	10.00
4	0.44	0.96	0.04
5	10.00	10.00	10.00
6	2.04	2.40	0.84
7	0.00	0.00	0.00
8	0.12	0.24	0.60
9	2.00	1.80	0.92
10	8.76	9.96	5.60
Total	53.36	55.36	48.00

TABLE V: Score of MiLSHADE-LSP with different *maxevals* values

only involves a greater number of evaluations, allowing to increase the accuracy of solutions, but also it could affect the search balance. A higher *maxevals* value gives more resources to achieve more accuracy in the search, but, at the same time, encourages a greater global exploration, that could be excessive to obtain good results.

Table V shows the scores by function and total for different values of *maxevals* ($10 \cdot 10^4 \cdot dim$, $25 \cdot 10^4 \cdot dim$, and $50 \cdot 10^4 \cdot dim$). It can be observed that increasing the number of evaluations produces a better score, but the improvement is lower than expected, only two more points with more than double of evaluations. However, when $maxevals$ increases even more, results are worse, maybe due to a worse trade-off between exploration and exploitation. Thus, the fact that MiLSHADE-LSP has adaptive variables related to the ratio of evaluations, becomes our algorithm very sensitive to the $maxevals$ parameter, as expected.

Function /evaluations	$Maxevals_1$ $10 \cdot 10^4 \cdot dim$	$Maxevals_2$ $25 \cdot 10^4 \cdot dim$	(In %) $\frac{Evals\ Maxevals_1}{Evals\ Maxevals_2}$
f_1	307	307	100%
f_2	287444	276423	103%
f_3	72308	65767	109%
f_4	988132	2285193	43%
f_5	9812	8313	118%
f_6	125047	183564	64%
f_{10}	926102	2139961	43%
Total	2140640	6804199	35%

TABLE VI: Ratio of evaluations in optimized runs functions for different $maxevals$

In order to study the speed of convergence, Table VI shows the number of evaluations when the optima are calculated with different *maxevals*. The third column is the relative difference in evaluations to achieve the optima between lower maximum evaluation number ($10 \cdot 10^4 \cdot dim$) and with the greater evaluation number ($25 \cdot 10^4 \cdot dim$). A value lower than 100% means that the optima can be obtained with less evaluations when *maxevals* is lower. On the contrary, a value higher than 100% means that it is more efficient a greater *maxevals*. While in functions f_1, f_2, f_3 and f_5, the number of evaluations are lightly reduced when $maxevals$ increases, in functions f_4, f_6, and f_{10}, the number of evaluations greatly increases with a higher $maxevals$. Although a higher $maxevals$ increases lightly the score, makes the convergence slower. Thus, $maxevals = 10 \cdot 10^4 \cdot dim$ could be considered the most adequate value if speed is important.

Function	$maxevals$	I_{Step}
1	$10 \cdot 10^4 \cdot dim$	10%
2	$50 \cdot 10^4 \cdot dim$	10%
3	$10 \cdot 10^4 \cdot dim$	10%
4	$50 \cdot 10^4 \cdot dim$	5%
5	$50 \cdot 10^4 \cdot dim$	5%
6	$10 \cdot 10^4 \cdot dim$	20%
7	$25 \cdot 10^4 \cdot dim$	5%
8	$50 \cdot 10^4 \cdot dim$	10%
9	$10 \cdot 10^4 \cdot dim$	10%
10	$10 \cdot 10^4 \cdot dim$	5%

TABLE VII: Tuned Parameter Values

c) Tuned vs non-tuned MiLSHADE-LSP: The competition 100-digit allows researcher to tune 2 parameter(s) independently for each problem. Table VII shows the used tuned parameters in our case, $maxevals$ and I_{Step}. With this tuning we obtain the results shown in Table X. It can be seen that tuned version achieves a total score of 60.72, more than 7 points more than the non-tuned version. In particular, it is obtained the optima in all cases in one more function, f_{10}. Also, it is obtained a significant improvement in functions f_4, f_6 (it obtains the optimum 13 times instead of 5 times), and f_8 (it obtains two right digits 7 times instead of only once).

To summarize, the LS pool improves both the results and the performance. Moreover, an adequate $maxevals$ value is needed to obtained a good trade-off between exploration and exploitation. Finally, tuning $maxevals$ and I_{Step} parameters by function improves significantly the accuracy. In general, the proposed algorithm, MiLSHADE-LSP, obtains very good results in the 100-digit challenge.

IV. Conclusions

In this paper, we have proposed a new optimization algorithm, MiLSHADE-LSP, a memetic algorithm which combines an adaptive differential evolution algorithm that modifies its behavior during the run, with a LS Pool that continuously improves the results. MiLSHADE-LSP uses a pool of two very different LS methods, the L-BFGS-B and the Solis-Wets algorithms. A memory is used to adapt which LS method is selected in each iteration. The DE algorithm, iL-SHADE, is continuously applied to the same population among iterations, to introduce the performance obtained by the LS to guide the search. Also, it has a restart mechanism to explore new areas when the search is stuck, restarting the population, and resetting the LS Pool parameters.

In the experimental section, we have tested and analyzed MiLSHADE-LSP using the proposed benchmark for the competition *100-digit challenge on Single Objective Numerical Optimization*. First, we have compared the contribution of the memetic proposal over the original iL-SHADE, obtaining that it contributes to improve not only the results, but also its performance. Later, we have analyzed the behavior of the proposal when the maximum evaluation number is increased. We have obtained that, due to the adaptive parameters, a greater $maxevals$ could getting worse results. Also, even when a greater $maxevals$ increased lightly the results in score, the speed to reach the optimum was reduced in several functions. Finally, as it was allowed in the competition, we have tuned two parameters independently by each problem, $maxevals$ and the initial step size, I_{Step}. We have observed that tuned-version got better results, obtaining a total score of more than 60. Thus, MiLSHADE-LSP have achieved very good results in the benchmark for the competition.

As a future work, we are going to try to adapt the I_{Step}, and improve other components, like the restart mechanism, to improve even more the results.

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).

References

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

[2] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms," California Institute of Technology, Tech. Rep. 826, 1989.

[3] P. Moscato and M. Norman, "A competitive and cooperative approach to complex combinatorial search," Caltech Concurrent Computation Program, Tech. Rep. 790, 1989.

[4] Y. S. Ong and A. J. Keane, "Meta-lamarckian learning in memetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 2, pp. 99–110, April 2004.

[5] R. Tanabe and A. S. Fukunaga, "Improving the search performance of SHADE using linear population size reduction," in *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2014, Beijing, China, July 6-11, 2014*, 2014, pp. 1658–1665.

[6] J. Brest, M. S. Mauec, and B. Bokovi, "iL-SHADE: Improved L-SHADE algorithm for single objective real-parameter optimization," in *2016 IEEE Congress on Evolutionary Computation (CEC)*, July 2016, pp. 1188–1195.

[7] K. Price, N. Awad, M. Ali, and P. Suganthan, "Problem definitions and evaluation criteria for the 100-digit challenge special session and competition on single objective numerical optimization," Nanyang Technological University, Singapore, Tech. Rep., November, 2018.

[8] F. J. Solis and R. J. Wets, "Minimization by Random Search Techniques," *Mathematical Operations Research*, vol. 6, pp. 19–30, 1981.

[9] 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.

[10] A. Molina, D. LaTorre and F. Herrera, "SHADE with iterative local search for large-scale global optimization," in *IEEE Congress on Evolutionary Computation*, 2018, pp. 1252–1259.