A SHADE-Based Algorithm for Large Scale Global Optimization*

Oscar Pacheco-Del-Moral and Carlos A. Coello Coello[0000−0002−8435−680X]

CINVESTAV-IPN (Evolutionary Computation Group)
Computer Science Department
Av. IPN No. 2508, Col. San Pedro Zacatenco
México D.F. 07300, MEXICO
opacheco@computacion.cs.cinvestav.mx
ccoello@cs.cinvestav.mx

Abstract. During the last decade, large-scale global optimization has been a very active research area not only because of its many challenges but also because of its high applicability. It is indeed crucial to develop more effective search strategies to explore large search spaces considering limited computational resources. In this paper, we propose a new hybrid algorithm called *Global and Local search using Success-History Based Parameter Adaptation for Differential Evolution* (GL-SHADE) which was specifically designed for large-scale global optimization. Our proposed approach uses two populations that evolve differently allowing them to complement each other during the search process. One is in charge of exploring the search space while the other is in charge of exploiting it. Our proposed method is evaluated using the CEC'2013 large-scale global optimization (LSGO) test suite with 1000 decision variables. Our experimental results show that the new proposal outperforms one of the best hybrid algorithms available in the state of the art (SHADEILS) in the majority of the test problems adopted while being competitive with respect to several other state-of-the-art algorithms when using the LSGO competition criteria adopted at CEC'2019.

Keywords: Differential Evolution · SHADE · Large Scale · Global Optimization · Hybrid Algorithms

1 Introduction

The general (single-objective) global optimization problem is defined as follows[1]:

$$\begin{aligned} \text{Minimize} \quad & f(\boldsymbol{x}) \\ \text{Subject to} \quad & lb_j \leq x_j \leq ub_j, \quad j = 1, 2, ..., D \end{aligned} \tag{1}$$

* The first author acknowledges support from CONACyT and CINVESTAV-IPN to pursue graduate studies in Computer Science. The second author gratefully acknowledges support from CONACyT grant no. 2016-01-1920 (*Investigación en Fronteras de la Ciencia 2016*) and from a SEP-Cinvestav grant (application no. 4).

[1] Without loss of generality, we will assume minimization.

where $\boldsymbol{lb}, \boldsymbol{ub} \in \mathbb{R}^D$ are the lower bound and the upper bound of the decision variables $\boldsymbol{x}$, respectively. $f : \mathbb{R}^D \rightarrow \mathbb{R}$ is the objective function. The feasible solution space is defined as: $\Omega = \{\boldsymbol{x} \in \mathbb{R}^D | lb_j \leq x_j \leq ub_j, \forall j \in \{1, 2, 3, ..., D\}\}$. When $D \geq 1000$, this is called large scale global optimization (LSGO) [11] and because of the limitations of mathematical programming techniques (particularly when dealing with highly nonlinear objective functions [2, 17]), the use of metaheuristics (particularly evolutionary algorithms) has become relatively popular [5]. Differential Evolution (DE) [14, 1] is a metaheuristic designed for continuous search spaces that has been very successful in solving a variety of complex optimization problems. However, as happens with other meta-heuristics, the performance of DE quickly deteriorates as we increase the number of decision variables of the problem (the so-called "curse of dimensionality") [3]. Additionally, the properties and conditions of the fitness landscape may change (e.g., going from unimodal to multimodal) [2, 3].

Many current approaches for large-scale global optimization are based on cooperative coevolution (CC), but several non-CC have also been proposed in recent years [5, 11, 2, 12, 17, 4]. CC consists in decomposing the original large scale problem into a set of smaller subproblems which are easier to solve (i.e., it is a divide-and-conquer approach) [2, 12, 17]. On the other hand, the non-CC approaches try to solve the large scale problem as a whole. Most of these non-CC approaches are hybrid schemes that combine several metaheuristics that complement each other in order to overcome their limitations. In fact, several researchers [5, 11, 2] have combined DE with non-population-based local search methods to boost its overall performance.

In this paper, we propose a new non-CC approach which is based on the so-called *Success-History Based Parameter Adaptation for Differential Evolution* (SHADE) algorithm [15]. Our proposed algorithm consists of three stages: (1) initialization, (2) global search and (3) local search. During the initialization stage a gradient-free non-population-based local search method is applied to one of the best individuals generated in order to make an early enhancement. Afterwards, the global and local search stages are iteratively repeated one after another. Our proposed approach consists of two populations that collaborate with each other since the first population presents a search scheme specialized in exploration (thus carrying out the global search stage) and the second one presents a search engine specialized in exploitation (carrying out the local search stage). The first population evolves according to SHADE's algorithm and the second one according to a new developed SHADE's variant which we've named $\text{eSHADE}_{\text{ls}}$. The communication between these two populations is done via a simple migration protocol (happening when switching from the global to the local search stage or vice versa). Our proposal is considered a hybrid and non-CC algorithm since it combines an evolutionary algorithm with a local search method (used just once during the initialization stage) and solves the problem as a whole (decomposition never happens). The performance of our proposed algorithm, referred to as *Global and Local search using SHADE* (GL-SHADE), is evaluated on the CEC'2013 large-scale global optimization (LSGO) test suite.

The remainder of this paper is organized as follows. In Section 2, we provide the background required to understand the rest of the paper. Section 3 describes in detail our proposal. Our experimental design and our results are provided in Section 4. Finally, our conclusions and some paths for future research work are provided in Section 5.

2 Background

2.1 Differential Evolution

Differential Evolution (DE) was originally proposed by Storn and Price in 1995 [14, 18]. DE is a stochastic population-based evolutionary algorithm (EA) that has shown to perform well in a variety of complex (including some real-world) optimization problems [14, 1]. DE has three control parameters: NP (population size), F (mutation scale factor) and Cr (crossover rate) [3]. It is well known that the performance of DE is very sensitive to these parameters [1]. Like other EAs, an initialization phase is its first task [18]. After that, DE adopts three main operators: mutation, recombination, and selection.

Algorithm 1: Standard Differential Evolution variant $DE/rand/1/bin$ [14].

```
   Data: max_FEs , NP , Cr , F
   Result: x ∈ P such that f(x) ≤ f(y), ∀y ∈ P \ {x}
1  Create a population P randomly and uniformly distributed over Ω ;
2  current_FEs = 0 + NP;
3  while current_FEs < max_FEs do                                  // stopping criterion
4      for i = 0 to i < NP do                                      // for every individual in the population
5          Take r1, r2, r3 ∈ [0, NP − 1] randomly ;                // r1 ≠ r2 ≠ r3 ≠ i, rn ∈ ℕ
6          v_i = x_r1 + F ∗ (x_r2 − x_r3) ;                        // mutation: F ∈ [0.0, 2.0] and x_rn ∈ P
7          j_rand = randInt(0, D − 1) ;                            // j_rand ∈ ℕ
8          for j = 0 to j < D do                                   // starting binomial crossover
9              if flip(Cr) || j == j_rand then                     // Cr ∈ [0.0, 1.0]
10                 u_i,j = v_i,j ;
11                 If u_i,j out of boundary then get it back to the feasible region;
12             else
13                 u_i,j = x_i,j ;
14         P_i^next ← fittest between u_i and x_i ;   // selection: take u_i if f(u_i) ≤ f(x_i), x_i otherwise
15         current_FEs + +;
16     P ← P^next ;                                                // advance generation and repeat
17 return (x ∈ P such that f(x) ≤ f(y), ∀y ∈ P \ {x}) ;             // fittest
```

The initialization phase (see Algorithm 1, line 1) consists in randomly scattering NP guesses (points) over the search space as follows: $x_{i,j} = lb_j + (ub_j - lb_j) * rnd(0, 1)$, where $x_{i,j}$ represents the j^{th} gene of the i^{th} individual [18].

The mutation and recombination operators are applied to generate a new trial vector ($\boldsymbol{u_i}$). During mutation, DE creates a new obtained candidate solution called a *donor* solution ($\boldsymbol{v_i}$). The sort of mutation and recombination operator to be adopted is defined based on the DE version that we use. The notation $DE/\alpha/\beta/\gamma$ is adopted to indicate the particular DE variant to be adopted [1]: α specifies the base vector, β is the number of difference vectors used, and γ denotes the type of recombination to be used [3]. Algorithm 1 presents the classical DE variant, called $DE/rand/1/bin$. In this scheme, for the mutation operator, three mutually exclusive individuals ($\boldsymbol{x_{r1}}$, $\boldsymbol{x_{r2}}$ and $\boldsymbol{x_{r3}}$) are randomly selected where

$\boldsymbol{x_{r1}}$ is perturbed using the scaled difference between $\boldsymbol{x_{r2}}$ and $\boldsymbol{x_{r3}}$ in order to obtain $\boldsymbol{v_i}$ (lines 5-6). For the recombination operator, genes are inherited from $\boldsymbol{v_i}$ and from the target vector ($\boldsymbol{x_i}$) and a binomial (bin) distribution is adopted in order to obtain $\boldsymbol{u_i}$ (lines 7-13) where at least one gene must be inherited from $\boldsymbol{v_i}$ (see line 9). The last step is the selection operator (line 14) where the fittest between $\boldsymbol{u_i}$ and $\boldsymbol{x_i}$ is chosen to represent the next generation's $\boldsymbol{x_i}$.

Several DE variants exist (see [14, 1]) and the selection of any of them will influence the search behavior of DE in different ways. Since there are only two types of recombination (exponential (exp) and binomial (bin)) the mutation strategy is really the one that better identifies the behavior of a particular DE scheme [1].

2.2 An Enhanced Differential Evolution Algorithm Based on Multiple Mutation Strategies

The *Enhanced Differential Evolution Algorithm Based on Multiple Mutation Strategies* (abbreviated as EDE by its authors) [18] was proposed as an enhancement of the $DE/best/1/bin$ scheme, aiming to overcome the tendency of this scheme to present premature convergence. The core idea of EDE is to take advantage of the direction guidance information of the best individual produced by the $DE/best/1/bin$ scheme, while avoiding being trapped into a local optimum. In the EDE algorithm, an opposition-based learning initialization scheme is combined with a mutation strategy composed of two DE variants ($DE/current/1/bin$ and $DE/pbest/bin/1$) aiming to speed up convergence and to prevent DE from clustering around the global best individual. EDE also incorporates a perturbation scheme for further avoiding premature convergence. Algorithm 2 shows the way in which EDE works.

Algorithm 2: An Enhanced Differential Evolution Algorithm Based on Multiple Mutation Strategies [18].

Data: max_{FEs} , NP , Cr , F, M, r_{max}, r_{min}, w_{max}, w_{min}
Result: $\boldsymbol{x} \in P$ such that $f(\boldsymbol{x}) \leq f(\boldsymbol{y}), \forall \boldsymbol{y} \in P \setminus \{\boldsymbol{x}\}$

1 Create a population P using an opposition-based learning initialization technique;
2 Update $current_{FEs}$ accordingly;
3 **while** $current_{FEs} < max_{FEs}$ **do** // stopping criterion
  // Execute one generation of DE based on multiple mutation strategies
4 **for** $i = 0$ *to* $i < NP$ **do** // for every individual in the population
5 Sort P from best to worst and set $pbest = randInt(0, M-1)$;
6 Take $r_2, r_3 \in [0, NP-1]$ randomly ; // $r_2 \neq r_3 \neq i$, $r_n \in \mathbb{N}$
7 $r_1 = r_{max} - \frac{current_{FEs}}{max_{FEs}} * (r_{max} - r_{min})$; // $r_1 \in \mathbb{R}$
8 **if** $flip(r1)$ **then**
9 $\boldsymbol{v_i} = \boldsymbol{x_i} + F * (\boldsymbol{x_{r_2}} - \boldsymbol{x_{r_3}})$;
10 **else**
11 $\boldsymbol{v_i} = \boldsymbol{x_{pbest}} + F * (\boldsymbol{x_{r_2}} - \boldsymbol{x_{r_3}})$;
12 Apply binomial recombination ; // see Algorithm 1, lines 7-13
13 $P_i^{next} \leftarrow$ fittest between $\boldsymbol{u_i}$ and $\boldsymbol{x_i}$; // selection: take $\boldsymbol{u_i}$ if $f(\boldsymbol{u_i}) \leq f(\boldsymbol{x_i})$, $\boldsymbol{x_i}$ otherwise
14 $current_{FEs} + +$;
15 $P \leftarrow P^{next}$; // advance generation and repeat
  // Perturb the best (fittest) individual in the population, dimension by dimension
16 **for** $j = 0$ *to* $j < D$ **do**
17 $\boldsymbol{\mu} = \boldsymbol{x_{best}}$; // Copy the best chromosome so far to μ, $\boldsymbol{x_{best}}$
18 $k = rnd(0, NP-1)$ such that $k \neq best$; // $k \in \mathbb{N}$
19 $n = rnd(0, D-1)$ such that $n \neq j$; // $n \in \mathbb{N}$
20 $r_2 = w_{min} + \frac{current_{FEs}}{max_{FEs}} * (w_{max} - w_{min})$; // $r_2 \in \mathbb{R}$
21 **if** $flip(r2)$ **then**
22 $\boldsymbol{\mu_j} = \boldsymbol{x_{best,n}} + (2 * rndreal(0,1) - 1) * (\boldsymbol{x_{best,n}} - \boldsymbol{x_{k,n}})$; // $\boldsymbol{x_k} \in P$
23 **else**
24 $\boldsymbol{\mu_j} = \boldsymbol{x_{best,j}} + (2 * rndreal(0,1) - 1) * (\boldsymbol{x_{best,n}} - \boldsymbol{x_{k,n}})$;
25 If $\boldsymbol{\mu_j}$ out of boundary then get it back to the feasible region ;
26 Evaluate the new chromosome $\boldsymbol{\mu}$ using f and increment $current_{FEs}$;
27 Take the fittest, between $\{\boldsymbol{\mu}, \boldsymbol{x_{best}}\}$ to represent $\boldsymbol{x_{best}}$;
28 **return** ($\boldsymbol{x} \in P$ *such that* $f(\boldsymbol{x}) \leq f(\boldsymbol{y}), \forall \boldsymbol{y} \in P \setminus \{\boldsymbol{x}\}$) ; // fittest

A remarkable feature of EDE is that it first uses a very explorative evolutionary strategy, but as the number of function evaluations increases, it changes to a much less explorative scheme (EDE sets $r_{min} = 0.1$ and $r_{max} = 1$). Another remarkable feature of EDE is the coupled perturbation method that it adopts between generations. Actually, if no mutation strategy switching is incorporated and we decide to use only line 11 (i.e., the mutation operation where $M = 1$), the procedure is transformed into a pure population-based local search method.

2.3 Success-History Based Parameter Adaptation for Differential Evolution

The *Success-History Based Parameter Adaptation for Differential Evolution* (SHADE) [15] is a self-adaptive version of the DE variant called $DE/current\text{-}to\text{-}pbest/1/bin$. The self-adaptive mechanism is applied to both F and Cr. Therefore, NP is its only control parameter (for further information about the self-adaptive mechanism of SHADE, interested readers are referred to [15]). SHADE also incorporates an external archive (A) built from the defeated parents throughout generations, since such individuals are used during the application of the mutation operator in order to promote a greater diversity. The adopted mutation strategy is the following:

$$\boldsymbol{v_i} = \boldsymbol{x_i} + F_i * (\boldsymbol{x_{pbest}} - \boldsymbol{x_i}) + F_i * (\boldsymbol{x_{r_2}} - \boldsymbol{x_{r_3}}) \tag{2}$$

where F is regenerated for every $x \in P$ (the regeneration procedure can be consulted in [15]). x_{r_2} and x_{r_3} are randomly chosen vectors from P and $P \cup A$, respectively, and x_{pbest} is taken randomly from the p% fittest individuals in the population. The parameter p is regenerated for every $x \in P$ as follows:

$$p_i = rand(\frac{2}{NP}, p_{max}) \tag{3}$$

where $p_{max} = 0.2$. The mutation strategy described in equation (2) is explorative as the base vector is practically the target vector. Another important feature is that information about the fittest individuals are taken into account. Finally, binomial recombination is used, but Cr is regenerated in a way analogous to the F regeneration procedure.

3 Our Proposal

Our proposed approach is called *Global and Local Search using SHADE* (GL-SHADE), and its main procedure is illustrated in Algorithm 4. GL-SHADE consists of three main components (lines 1-3):

- **MTS-LS1** (stands for *Multiple Trajectory Search - Local Search 1*) is a gradient-free non-population-based local search method (for a detailed explanation of this method, interested readers should refer to [16]). This method is used at the beginning of the search (line 9) and just once in all the procedure (the idea is to boost the search at the beginning).

- **SHADE** is used for population 1 (line 1). It integrates a global search scheme which takes into account information about the top individuals in the population (this property is remarkable since all enhancements done by other methods to the fittest individual can be used to guide the search) and presents a robust self-adaptive mechanism. Due to the aforementioned features, this scheme is adopted to handle the global search of our proposed approach.
- **eSHADE$_{\mathbf{ls}}$** is used for population 2 (line 2). This is the variant called $SHADE/pbest/1/exp$ coupled with the EDE perturbation method (see Algorithm 2, lines 16-27). This variant uses the following mutation strategy:

$$\boldsymbol{v_i} = \boldsymbol{x_{pbest}} + F_i * (\boldsymbol{x_{r_2}} - \boldsymbol{x_{r_3}}) \tag{4}$$

where p_{max} is set to 0.1 (see equation (3)). The strategy described in equation (4) is one of the mutation operators that EDE incorporates for scattering new trial vectors close to several top individuals and not just near the fittest one. As can be seen, this strategy is very different to that of equation (2), and our proposed approach allows them to complement each other. Additionally, exponential recombination is adopted instead of binomial recombination. The main procedure of eSHADE$_{\text{ls}}$, which was designed to handle the local search, is described in Algorithm 3.

Algorithm 3: eSHADE-ls

```
   Data: L_FEs , NP_2 , w_min, w_max
1  Initialize population and all required parameters ;          // initialization
2  counter_FEs = 0 ;
3  while counter_FEs < L_FEs do                                 // start evolution
4  |  Execute one generation of SHADE/pbest/1/exp;
5  |  Apply the EDE perturbation method to the fittest individual in the population;
6  |  Update counter_FEs accordingly;
7  return (fittest individual so far) ;                         // end evolution
```

The first task in Algorithm 4 is an initialization stage (lines 1-9). During this stage, the GL-SHADE's components are defined (lines 1-3), the corresponding populations are generated (line 5), the fittest individual from population 1 is set as the best solution so far (line 7) and a local search procedure is applied to the best solution recorded so far (line 9); in this case, the MTS-LS1 method is used. When defining the components, it is necessary to provide the stopping criterion (maximum number of evaluations max_{FEs}) since in some cases, it is required to stop the overall search process even when the maximum number of evaluations hadn't been reached (G_{FEs} or L_{FEs}). For example, MTS-LS1's execution (line 9) stops when $counter_{FEs} \geq L_{FEs}$ or $current_{FEs} \geq max_{FEs}$. In fact, $counter_{FEs}$ and $current_{FEs}$ are updated accordingly as the component's execution goes by.

The second task is to perform a global and local search stage (lines 10-16). During this stage, population 1 receives (line 11) the new best individual (which must be placed at the position where the old best individual is) and then the global search scheme is executed (line 12) while $counter_{FEs} \leq G_{FEs}$ and

$current_{FEs} \leq max_{FEs}$. After finishing, population 1 migrates (lines 13-14) its best individual to population 2 (the entry individual is randomly placed) and then the local search scheme is executed (line 15) while $counter_{FEs} \leq L_{FEs}$ and $current_{FEs} \leq max_{FEs}$. Finally, population 2 migrates (lines 16-11) its best individual to population 1 and the procedure is repeated until the maximum number of function evaluations (max_{FEs}) is reached.

It is worth noting that after applying the mutation or perturbation procedures, a variable may fall outside its allowable boundaries. If this happens, we apply the same normalization procedure adopted in [15].

Algorithm 4: GL-SHADE

Data: max_{FEs}, G_{FEs}, L_{FEs}, NP_1, NP_2, $f_{objective}$, w_{min}, w_{max}
// Define components, set maximum number of evaluations and set the objective function
1 $DE_1 \leftarrow SHADE(NP_1, f_{objective}, max_{FEs})$;
2 $DE_2 \leftarrow eSHADE_{ls}(NP_2, f_{objective}, max_{FEs}, w_{min}, w_{max})$;
3 $LS_1 \leftarrow MTS_{\text{LS1}}(f_{objective}, max_{FEs})$;
4 $current_{FEs} = 0$; // keep track of the number of evaluations computed so far
// Initialize: populations, parameters and structures needed by the corresponding component
5 $DE_1.initialize()$, $DE_2.initialize()$, $LS_1.initialize()$;
6 $current_{FEs} = current_{FEs} + NP_1 + NP_2$; // update evaluations computed so far
7 $best_{global} \leftarrow DE_1.best$; // update best global
// Early local search: at the end of this stage $current_{FEs} += L_{FEs}$
8 $counter_{FEs} = 0$; // restart counter
9 $LS_1.enhance(best_{global}, L_{FEs}, counter_{FEs}, current_{FEs})$;
// While the number of evaluations is less than the maximum defined, do ...
10 **while** $current_{FEs} < max_{FEs}$ **do**
// Global search: at the end of this stage $current_{FEs} += G_{FEs}$
11 $DE_1.receive(best_{global})$ and $counter_{FEs} = 0$; // migrate better solution: from pop_2 to pop_1
12 $DE_1.evolve(G_{FEs}, counter_{FEs}, current_{FEs})$; // GS
13 $best_{global} \leftarrow DE_1.best$; // update best global
// Local search: at the end of this stage $current_{FEs} += L_{FEs}$
14 $DE_2.receive(best_{global})$ and $counter_{FEs} = 0$; // migrate better solution: from pop_1 to pop_2
15 $DE_2.evolve(L_{FEs}, counter_{FEs}, current_{FEs})$; // LS
16 $best_{global} \leftarrow DE_2.best$; // update best global
17 **Report** $best_{global}$

4 Experimental results

In order to assess the performance of our proposed GL-SHADE, we adopted the test suite used at the large-scale global optimization competition held at the *2013 IEEE Congress on Evolutionary Computation (CEC'2013)* [6], but adopting the experimental conditions and guidelines of the LSGO competition held at the *2019 IEEE Congress on Evolutionary Computation (CEC'2019)* [7].

The previously indicated benchmark consists of 15 test problems, all with 1000 decision variables except for $F13$ and $F14$ which are overlapping functions, where $D = 905$ [2, 11]. In general, these test problems can be categorized as shown in Table 1.

For each test problem, 25 independent executions were carried out. Each run is stopped when reaching a pre-defined maximum number of objective func-

tion evaluations ($max_{FEs} = 3 \times 10^6$). Additionally, we also report the results obtained after performing 120,000 and 600,000 objective function evaluations. The parameter values adopted in our experiments are shown in Table 2 and our results are summarized in Table 3 (we report the best, worst, median, mean, and standard deviations calculated over the 25 runs performed for each test problem).

Table 1: Features of the test problems from the CEC'2013 benchmark

Category	Functions
Fully separable	F1-F3
Partially separable	F4-F11
Overlapping	F12-F14
Fully non-separable	F15

Table 2: Parameter values used in our experiments

Parameter	Value	Usage
NP_1	100	Size of Population 1
NP_2	100	Size of Population 2
G_{FEs}	25000	Max evaluations requested for global search per iteration
L_{FEs}	25000	Max evaluations requested for local search per iteration
w_{min}	0.0	$\text{eSHADE}_{\text{ls}}$ algorithm
w_{max}	0.2	$\text{eSHADE}_{\text{ls}}$ algorithm

Table 3: Summary of the results obtained by GL-SHADE in the CEC'2013 test problems

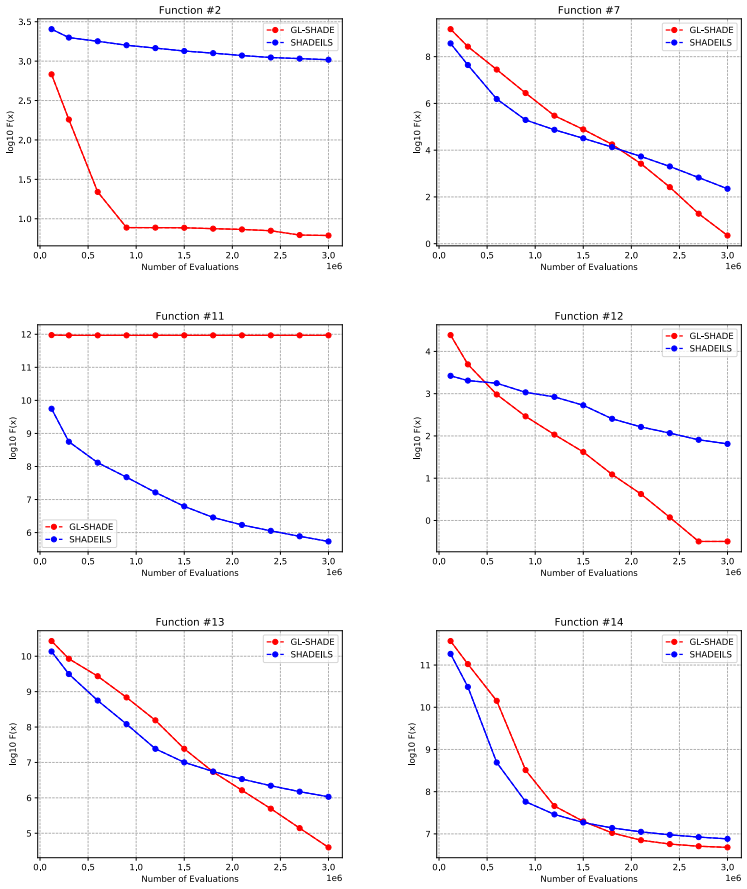
1000D		*f1*	*f2*	*f3*	*f4*	*f5*	*f6*	*f7*	*f8*
	Best	1.0922E+05	6.0793E+02	2.0003E+01	2.6578E+10	3.2918E+06	1.0475E+06	4.5536E+08	1.1335E+14
	Median	1.6125E+05	6.8597E+02	2.0003E+01	5.4221E+10	4.5740E+06	1.0547E+06	1.3964E+09	4.4679E+14
1.2E+05	Worst	4.6460E+05	7.8329E+02	2.0004E+01	9.4616E+10	5.6865E+06	1.0597E+06	3.0432E+09	8.5885E+14
	Mean	1.7899E+05	6.8888E+02	2.0003E+01	5.4236E+10	4.5864E+06	1.0546E+06	1.4949E+09	4.4115E+14
	StDev	7.1896E+04	4.1088E+01	3.5612E-04	1.7031E+10	5.8026E+05	3.6927E+03	6.2989E+08	1.9130E+14
	Best	7.2085E-04	1.4647E+01	2.0000E+01	1.0450E+09	1.9819E+06	1.0474E+06	1.2055E+07	1.6256E+12
	Median	3.9288E+01	2.0938E+01	2.0000E+01	2.5779E+09	2.7543E+06	1.0544E+06	2.2112E+07	9.0666E+12
6.0E+05	Worst	1.4477E+02	3.8803E+01	2.0000E+01	5.2019E+09	3.6524E+06	1.0596E+06	8.1126E+07	4.9873E+13
	Mean	4.5649E+01	2.1808E+01	2.0000E+01	2.6494E+09	2.6890E+06	1.0541E+06	2.9485E+07	1.2236E+13
	StDev	4.1617E+01	6.1271E+00	2.4000E-05	1.0419E+09	4.2647E+05	3.5944E+03	1.6787E+07	1.1786E+13
	Best	2.5836E-26	9.9496E-01	2.0000E+01	9.3076E+06	1.3728E+06	1.0107E+06	4.5790E-02	3.9375E+09
	Median	9.4995E-26	4.9748E+00	2.0000E+01	2.4488E+07	2.1891E+06	1.0348E+06	1.0654E+00	3.3649E+10
3.0E+06	Worst	8.7787E-23	2.7859E+01	2.0000E+01	7.8810E+07	2.8653E+06	1.0534E+06	1.0269E+01	3.2976E+11
	Mean	1.0930E-23	6.6067E+00	2.0000E+01	2.7387E+07	2.2180E+06	1.0342E+06	2.1701E+00	8.9404E+10
	StDev	2.3448E-23	6.9108E+00	0.0000E+00	1.5706E+07	3.6639E+05	1.0878E+04	2.5127E+00	1.1417E+11
1000D		*f9*	*f10*	*f11*	*f12*	*f13*	*f14*	*f15*	
	Best	1.2351E+09	9.2531E+07	9.2224E+11	1.9287E+04	1.5994E+10	9.3400E+10	9.0004E+07	
	Median	2.3031E+09	9.4008E+07	9.3948E+11	2.3414E+04	2.6080E+10	3.4485E+11	1.1240E+08	
1.2E+05	Worst	5.7137E+09	9.4342E+07	1.0020E+12	3.0089E+04	4.8688E+10	7.6595E+11	1.4611E+08	
	Mean	2.3688E+09	9.3877E+07	9.4278E+11	2.3987E+04	2.6841E+10	3.7238E+11	1.1368E+08	
	StDev	9.3671E+08	4.4457E+05	1.7790E+10	3.4976E+03	7.3622E+09	1.7564E+11	1.4759E+07	
	Best	1.6273E+09	9.2088E+07	9.1590E+11	1.7099E+02	1.2233E+09	5.3650E+08	9.2158E+06	
	Median	2.1679E+09	9.2980E+07	9.2092E+11	8.9949E+02	2.9356E+09	8.8536E+09	3.2383E+07	
6.0E+05	Worst	4.7715E+09	9.4153E+07	9.4439E+11	1.8122E+03	3.9737E+09	3.0699E+10	6.9700E+07	
	Mean	2.3619E+09	9.3006E+07	9.2554E+11	9.2076E+02	2.7333E+09	1.0825E+10	3.1321E+07	
	StDev	7.3543E+08	4.6799E+05	1.0055E+10	4.3518E+02	8.1207E+08	8.8037E+09	1.7574E+07	
	Best	1.3860E+09	9.0964E+07	9.1543E+11	1.2587E-23	1.3388E+04	4.3995E+06	1.2041E+05	
	Median	2.1147E+09	9.1681E+07	9.2276E+11	2.0667E-23	2.9806E+04	4.7472E+06	1.1409E+06	
3.0E+06	Worst	3.6649E+09	9.2675E+07	9.4943E+11	3.9866E+00	1.2446E+05	5.3146E+06	3.0018E+06	
	Mean	2.1871E+09	9.1750E+07	9.2729E+11	7.9732E-01	3.9831E+04	4.7868E+06	1.2919E+06	
	StDev	6.3160E+08	4.7189E+05	1.0580E+10	1.6275E+00	2.9868E+04	2.1478E+05	1.1058E+06	

Finally, we also provide the convergence curves for $f2$, $f7$, $f11$, $f12$, $f13$ and $f14$ (see Fig. 1). In order to reduce its running time, GL-SHADE and the benchmark set were implemented[2] using C++ and CUDA. All experiments were

[2] Our source code can be obtained from: https://github.com/delmoral313/gl-shade

performed using the Intel(R) Core(TM) i7-3930K CPU @ 3.20GHz with 8 GB RAM (using the Ubuntu 18.04 operating system), and the GeForce GTX 680 GPU with the CUDA 10.2 version.

Fig. 1: Convergence curves with logarithmic scale for some CEC'2013 benchmark problems



4.1 Comparison with its components

In order to investigate if GL-SHADE is able to outperform its individual components, we adopted the Wilcoxon signed-rank test. Here, we take $N = 25$ and $\alpha = 0.05$, meaning that we use a sample size of 25 executions and a significance level of 5%. Results are summarized in Table 4. There, we can see the mean

as well as its statistical significance next to it. The notation "b/e/w" shown in the last row means that a component is significantly better in "b" test problems, that there was no statistically significant difference in "e" test problems and that it was significantly worse in "w" test problems, with respect to GL-SHADE. These results indicate that our proposed approach presents a better overall performance than any of its components considered separately. This is particularly true for the overalapping and partially separate test problems with a separable subcomponent (i.e., $f4$ to $f7$).

Table 4: Statistical validation (GL-SHADE is the control algorithm).

GL-SHADE vs. its components				
Function	**GL-SHADE**	**MTS-LS1**	**SHADE**	**eSHADE-ls**
$f1$	1.0930e-23	3.0182E-25 $\approx$	2.1845E+06 $-$	1.8548E-16 $-$
$f2$	6.6067e+00	4.0137E+03 $-$	1.5379E+04 $-$	3.3829E+00 $\dagger$
$f3$	2.0000e+01	2.0007E+01 $-$	2.0060E+01 $-$	2.0000E+01 $\approx$
$f4$	2.7387e+07	1.0573E+12 $-$	1.1522E+09 $-$	2.3669E+08 $-$
$f5$	2.2180e+06	5.9109E+07 $-$	2.3572E+06 $\approx$	1.1502E+07 $-$
$f6$	1.0342e+06	1.0506E+06 $-$	1.0564E+06 $-$	1.0389E+06 $\approx$
$f7$	2.1701e+00	7.9165E+09 $-$	3.1042E+06 $-$	3.1663E+02 $-$
$f8$	8.9404e+10	5.5029E+16 $-$	8.9442E+11 $-$	5.4067E+13 $-$
$f9$	2.1871e+09	4.3869E+10 $-$	1.3792E+09 $\dagger$	2.3750E+09 $\approx$
$f10$	9.1750e+07	9.4198E+07 $-$	9.2711E+07 $-$	9.2001E+07 $\approx$
$f11$	9.2729e+11	1.1674E+12 $-$	9.3339E+11 $\approx$	9.3258E+11 $\approx$
$f12$	7.9732e-01	2.1449E+03 $-$	8.2921E+06 $-$	3.6372E+02 $-$
$f13$	3.9831e+04	4.0259E+10 $-$	5.7585E+07 $-$	2.9525E+05 $-$
$f14$	4.7868e+06	1.1431E+12 $-$	1.2944E+08 $-$	4.9800E+06 $\approx$
$f15$	1.2919e+06	2.9521E+08 $-$	1.4229E+06 $\approx$	5.8184E+05 $\dagger$
b/e/w		**0/1/14**	**1/3/11**	**2/6/7**

4.2 Comparison between GL-SHADE and SHADEILS

SHADE with iterative local search (SHADEILS) [11] is one of the best hybrid algorithms currently available for large-scale global optimization, according to [9, 10]. SHADEILS uses SHADE to handle the global search and it incorporates MTS-LS1 and a gradient-based method to undertake the local search. Additionally, it integrates a re-start mechanism which is launched when stagnation is detected. In order to compare SHADEILS with respect to our proposal, we adopted again the test suite used at the large-scale global optimization competition held at the *2013 IEEE Congress on Evolutionary Computation (CEC'2013)* [6]. We also applied the Wilcoxon signed-rank test with $N = 25$ and $\alpha = 0.05$. Our comparison of results is summarized in Table 5. The results that are better in a statistically significant way are shown in **boldface**. Additionally, we show the convergence curves for both SHADEILS and GL-SHADE in Fig. 1. SHADEILS is significantly better than our proposed approach in 4 out of 15 test problems, while GL-SHADE is significantly better than SHADEILS in 9 out of 15 test problems. Our proposed approach is particularly better than SHADEILS in the overlapping test problems.

Table 5: Statistical comparison of results: SHADEILS vs GL-SHADE using the CEC'2013 benchmark problems (SHADEILS is the control algorithm), performing 3,000,000 objective function evaluations

	Mean ±Std. Dev.		
Function	**SHADEILS**	**GL-SHADE**	**Sig.**
$f1$	**2.5558E-28 ±5.3619E-28**	1.0930E-23 ±2.3448E-23	–
$f2$	1.0415E+03 ±1.0341E+02	**6.6067E+00 ±6.9108E+00**	†
$f3$	2.0068E+01 ±4.7610E-02	**2.0000E+01 ±0.0000E+00**	†
$f4$	3.0128E+08 ±1.0458E+08	**2.7387E+07 ±1.5706E+07**	†
$f5$	**1.3310E+06 ±2.2657E+05**	2.2180E+06 ±3.6639E+05	–
$f6$	1.0316E+06 ±9.8658E+03	1.0342E+06 ±1.0878E+04	≈
$f7$	2.2356E+02 ±2.5286E+02	**2.1701E+00 ±2.5127E+00**	†
$f8$	5.9937E+11 ±5.4287E+11	**8.9404E+10 ±1.1417E+11**	†
$f9$	**1.5780E+08 ±1.4888E+07**	2.1871E+09 ±6.3160E+08	–
$f10$	9.2556E+07 ±4.5100E+05	**9.1750E+07 ±4.7189E+05**	†
$f11$	**5.3888E+05 ±2.2303E+05**	9.2729E+11 ±1.0580E+10	–
$f12$	6.4886E+01 ±2.2987E+02	**7.9732E-01 ±1.6275E+00**	†
$f13$	1.0706E+06 ±8.6269E+05	**3.9831E+04 ±2.9868E+04**	†
$f14$	7.6280E+06 ±1.2756E+06	**4.7868E+06 ±2.1478E+05**	†
$f15$	8.6832E+05 ±6.3444E+05	1.2919E+06 ±1.1058E+06	≈
b/e/w	**4/2/9**	**9/2/4**	

4.3 Comparison with respect to state-of-the-art algorithms

Our proposed approach was also compared with respect to three state-of-the-art algorithms (besides SHADEILS), namely:

- **CC-RDG3** [12]: is a CC-based algorithm. It was the winner of the 2019 LSGO competition [13, 10].
- **MOS** [5]: is a hybrid algorithm that was considered as one of the best metaheuristics for LSGO during several years (from 2013 to 2018) [11, 13].
- **LSHADE-SPA** [2]: is a hybrid-CC method. This approach together with SHADEILS were the first to outperform MOS [9].

For comparing our results, we adopted the 2019 LSGO competition criteria [10]. Thus, each algorithm is assigned a score (per test problem) using the *Formula One Score* (FOS) which is based on its position (1st/25pts, 2nd/18pts, 3rd/15pts, 4th/12pts and 5th/10pts). The comparison of results (performing 3,000,000 objective function evaluations) of our proposed GL-SHADE with respect to other state-of-the-art algorithms is shown in Fig. 2. The results from the other algorithms were extracted from the LSGO competition database [13, 7, 8]. Based on the results summarized in Fig. 2, we obtained the ranking shown in Table 6.

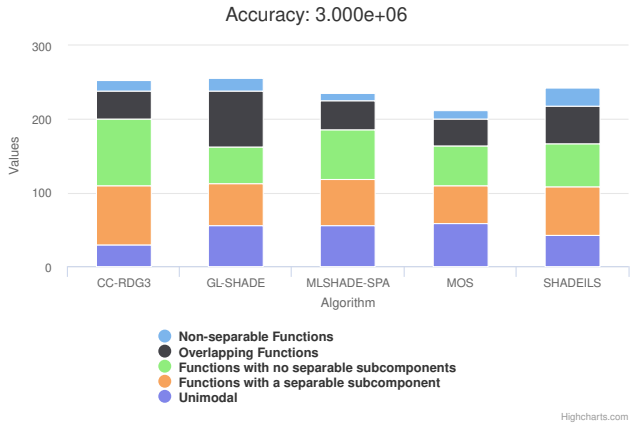
5 Conclusions and Future Work

In this paper, we proposed a new DE-based algorithm (GL-SHADE) specifically designed to solve LSGO problems. Our proposed approach incorporates a global search engine combined with a local search mechanism. For the global search

Table 6: Ranking according to the FOS criterion

#	Algorithm	Type	FOS
1	*GL-SHADE*	Hybrid	256
2	*CC-RDG3*	CC	253
3	*SHADE-ILS*	Hybrid	243
4	*MLSHADE-SPA*	CC-Hybrid	236
5	*MOS*	Hybrid	212

Fig. 2: GL-SHADE vs state-of-the-art algorithms using the CEC'2013 benchmark problems adopting the FOS criterion



engine, our proposed approach adopts SHADE. For our local search mechanism, we adopted a population-based self-adaptive algorithm (eSHADE-ls) which is indeed the main difference of our proposal with respect to other state-of-the-art algorithms adopted for LSGO. SHADE and eSHADE-ls collaborate with each other during the evolutionary process. Since our proposed approach adopts very different and complementary mutation strategies, SHADE's strategy scatters a mutated vector around the target vector, while eSHADE-ls' strategy scatters it close to one of the top individuals in the population. Our proposed approach was able to outperform its components (when considered independently), as well as SHADEILS in most of the test problems adopted (we adopted the CEC'2013 LSGO test problems). Additionally, it was found to be very competitive with respect to four state-of-the-art algorithms and obtained the best rank (based on the FOS criterion). As part of our future work, we are interested in experimenting with different hybrid schemes. For example, one possibility would be to have an automatic resource allocation mechanism that can determine the maximum number of evaluations that each component of our approach should be executed, with the aim of improving its performance. We are also interested in trying other (more elaborate) local search schemes that can also improve the performance of our proposed approach.

References

1. Das, S., Suganthan, P.N.: Differential Evolution: A Survey of the State-of-the-Art. IEEE Transactions on Evolutionary Computation **15**(1), 4–31 (February 2011)
2. Hadi, A.A., Mohamed, A.W., Jambi, K.M.: LSHADE-SPA memetic framework for solving large-scale optimization problems. Complex & Intelligent Systems **5**, 25–40 (March 2019)
3. Hiba, H., El-Abd, M., Rahnamayan, S.: Improving SHADE with Center-based Mutation for Large-scale Optimization. In: 2019 IEEE Congress on Evolutionary Computation (CEC'2019). pp. 1533–1540. IEEE, Wellington, New Zealand (10-13 June 2019)
4. Jian, J.R., Zhan, Z.H., Zhang, J.: Large-scale evolutionary optimization: a survey and experimental comparative study. International Journal of Machine Learning and Cybernetics **11**, 729–745 (March 2020)
5. LaTorre, A., Muelas, S., Peña, J.M.: Large Scale Global Optimization: Experimental Results with MOS-based Hybrid Algorithms. In: 2013 IEEE Congress on Evolutionary Computation (CEC 2013). pp. 2742–2749. IEEE, Cancún, México (20-23 June 2013)
6. Li, X., Tang, K., Omidvar, M.N., Yang, Z., Qin, K.: Benchmark Functions for the CEC'2013 Special Session and Competition on Large-Scale Global Optimization (2013)
7. Molina, D., LaTorre, A.: Toolkit for the Automatic Comparison of Optimizers: Comparing Large-Scale Global Optimizers Made Easy. In: 2018 IEEE Congress on Evolutionary Computation (CEC'2018). IEEE, Rio de Janeiro, Brazil (8-13 July 2018), iSBN 978-1-5090-6018-4
8. Molina, D., LaTorre, A.: Toolkit for the Automatic Comparison of Optimizers (TACO): Herramienta *online* avanzada para comparar metaheurísticas. In: XIII Congreso Español en Metaheurísticas y Algoritmos Evolutivos y Bioinspirados. pp. 727–732 (2018)
9. Molina, D., LaTorre, A.: WCCI 2018 Large-Scale Global Optimization Competition Results (2018), http://www.tflsgo.org/download/comp2018_slides.pdf, Last accessed on 2020-02-29
10. Molina, D., LaTorre, A.: CEC 2019 Large-Scale Global Optimization Competition Results (2019), http://www.tflsgo.org/assets/cec2019/comp2019_slides.pdf, Last accessed on 2020-02-29
11. Molina, D., LaTorre, A., Herrera, F.: SHADE with Iterative Local Search for Large-Scale Global Optimization. In: 2018 IEEE Congress on Evolutionary Computation (CEC'2018). IEEE, Rio de Janeiro, Brazil (8-13 July 2018)
12. Omidvar, M.N., Li, X., Mei, Y., Yao, X.: Cooperative Co-Evolution With Differential Grouping for Large Scale Optimization. IEEE Transactions on Evolutionary Computation **18**(3), 378–393 (June 2013)
13. Omidvar, M.N., Sun, Y., La Torre, A., Molina, D.: Special Session and Competition on Large-Scale Global Optimization on WCCI 2020 (2020), http://www.tflsgo.org/special_sessions/wcci2020.html, Last accessed on 2020-02-22
14. Storn, R., Price, K.: Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. Journal of Global Optimization **11**(4), 341–359 (December 1997)
15. Tanabe, R., Fukunaga, A.: Success-history based parameter adaptation for Differential Evolution. In: 2013 IEEE Congress on Evolutionary Computation (CEC'2013). pp. 71–78. IEEE, Cancún, México (20-23 June 2013)

16. Tseng, L.Y., Chen, C.: Multiple Trajectory Search for Large Scale Global Optimization. In: 2008 IEEE Congress on Evolutionary Computation (CEC'2008). pp. 3052–3059. IEEE, Hong Kong (1-6 June 2008)
17. Wu, X., Wang, Y., Liu, J., Fan, N.: A New Hybrid Algorithm for Solving Large Scale Global Optimization roblems. IEEE Access **7**, 103354–103364 (29 July 2019)
18. li Xiang, W., lei Meng, X., qing An, M., Li, Y., xia Gao, M.: An Enhanced Differential Evolution Algorithm Based on Multiple Mutation Strategies. Computational Intelligence and Neuroscience **2015** (2015), article ID: 285730