A lightweight SHADE-based algorithm for global optimization - liteSHADE

Citation

VIKTORIN, Adam, Roman ŠENKEŘÍK, Michal PLUHÁČEK, Tomáš KADAVÝ, and Roman JAŠEK. A lightweight SHADE-based algorithm for global optimization - liteSHADE. In: *Lecture Notes in Electrical Engineering* [online]. vol. 554, Springer Verlag, 2020, p. 197 - 206 [cit. 2023-02-02]. ISBN 978-3-03-014906-2. ISSN 1876-1100. Available at https://link.springer.com/chapter/10.1007/978-3-030-14907-9_20

DOI

https://doi.org/10.1007/978-3-030-14907-9_20

Permanent link

https://publikace.k.utb.cz/handle/10563/1008756

A Lightweight SHADE-Based Algorithm for Global Optimization – liteSHADE

Adam Viktorin, Roman Senkerik , Michal Pluhacek , Tomas Kadavy , and Roman Jasek

Faculty of Applied Informatics, Tomas Bata University in Zlin, T. G. Masaryka 5555, 760 01 Zlin, Czech Republic

{aviktorin,senkerik,pluhacek,kadavy,jasek}@utb.cz

Abstract

In this paper, a novel lightweight version of the Successful-History based Adaptive Differential Evolution (SHADE) is presented as the first step towards a simple, user-friendly, metaheuristic algorithm for global optimization. This simplified algorithm is called liteSHADE and is compared to the original SHADE on the CEC2015 benchmark set in three dimensional settings - 10D, 30D and 50D. The results support the idea, that simplification may lead to a successful and understandable algorithm with competitive or even better performance.

Keywords: Differential Evolution, SHADE, simplification, liteSHADE

1 Introduction

The Differential Evolution algorithm (DE) was firstly described by Storn and Price in 1995 [**1**] and since then, it has been thoroughly studied by a large number of scientists around the world. Selected examples of studies in this field might be found in one of the recent surveys [**2-4**].

The DE is primarily an algorithm for global optimization in continuous spaces and is based, as all other evolutionary computational techniques, on the Darwinian theory of evolution. It is also very simple and requires a setting of only three control parameters - population size NP, crossover rate CR and scaling factor F. While the number of control parameters is quite small, the influence of their setting to the performance of the algorithm is quite significant and therefore, they require fine-tuning [**5, 6**]. In order to avoid that task, recent DE variants incorporate self-adaptation of some of these parameters. The self-adaptation is based on the progress of optimisation of a given problem and therefore, it mostly adapts CR and F values, whereas the population size is not affected. Probably a most important recent self-adaptive DE variant is called Success-History based Differential Evolution (SHADE) and it was created by Tanabe and Fukunaga in 2013 [**7**]. The latest CEC single-objective competition winners were all based on this algorithm - 2014 L-SHADE [**8**], 2015 SPS-L-SHADE-EIG [**9**], 2016 LSHADE_EpSin [**10**] and 2017 jSO [**11**]. Thus, it is selected as a basis for a simplification in this paper.

According to a recent paper [**12**], some of the SHADE-based algorithms use unnecessarily complicated mechanisms and should be simplified to made them more clear to their users and might be even improved by such simplification. In this paper, a lightweight SHADE variant (liteSHADE) is presented and it shows some interesting future directions for the simplification of DE-based self-adaptive algorithms. It is shown on the CEC2015 benchmark set, that its performance is competitive with the original SHADE and that it can even provide better results in higher dimensions. The liteSHADE algorithm also uses an updated version of a previously published distance based parameter adaptation [**13**], which was shown to be quite effective for premature convergence avoidance.

The rest of the paper is structured as follows: The next section describes DE, SHADE and liteSHADE algorithms. Section **3** covers experimental settings, Sect. **4** provides results and their discussion and the whole paper is concluded by Sect. **5**.

2 From DE to liteSHADE

In order to describe the liteSHADE, it is important to start from the DE by Storn and Price [**1**]. The canonical 1995 DE is based on the idea of evolution from a randomly generated set of solutions of the optimization task called population ***P***, which has a preset size of NP. Each individual (solution) in the population consists of a vector x of length D (each vector component corresponds to one attribute of the optimized task) and objective function valuef(**x**), which mirrors the quality of the solution. The number of optimized attributes D is often referred to as the dimensionality of the problem and such generated population ***P***, represent the first generation of solutions.

The individuals in the population are combined in an evolutionary manner to create improved offspring for the next generation. This process is repeated until the stopping criterion is met (either the maximum number of generations, or the maximum number of objective function evaluations, or the population diversity lower limit, or overall computational time), creating a chain of subsequent generations, where each following generation consists of better solutions than those in previous generations - a phenomenon called elitism.

The combination of individuals in the population consists of three main steps: Mutation, crossover and selection.

In the mutation, attribute vectors of selected individuals are combined in simple vector operations to produce a mutated vector **v**. This operation uses a control parameter - scaling factor F. In the crossover step, a trial vector **u** is created by selection of attributes either from mutated vector **v** or the original vector **x** based on the crossover probability given by a control parameter - crossover rate CR. And finally, in the selection, the quality f(u) of a trial vector is evaluated by an objective function and compared to the quality f(x) of the original vector and the better one is placed into the next generation.

From the basic description of the DE algorithm, it can be seen, that there are three control parameters, which have to be set by the user - population size NP, scaling factor F and crossover rate CR. It was shown in [**5, 6**], that the setting of these parameters is crucial for the performance of DE. Fine-tuning of the control parameter values is a time-consuming task and therefore, many state-of-the-art DE variants use self-adaptation to avoid this cumbersome task. This is also a case of SHADE algorithm proposed by Tanabe and Fukunaga in 2013 [**7**] and since it is used in this paper, the algorithm is described in more detail in the next section.

2.1 Shade

As aforementioned, SHADE algorithm was proposed with a self-adaptive mechanism of some of its control parameters to avoid their fine-tuning. Control parameters in question are scaling factor F and crossover rate CR. It is fair to mention, that SHADE algorithm is based on Zhang and Sanderson's JADE [**14**] and shares a lot of its mechanisms. The main difference is in the historical memories MF and MCR for successful scaling factor and crossover rate values with their update mechanism.

Following subsections describe individual steps of the SHADE algorithm: Initialization, mutation, crossover, selection and historical memory update.

Initialization

The initial population P is generated randomly and for that matter, a Pseudo-Random Number Generator (PRNG) with uniform distribution is used. Solution vectors x are generated according to the limits of solution space - lower and upper bounds Eq. (**1**).

$$x_{j,i} = U\left[lower_j, upper_j\right] \text{ for } j = 1, \ldots, D; i = 1, \ldots, NP, \tag{1}$$

where i is the individual index and j is the attribute index. The dimensionality of the problem is represented by D, and NP stands for the population size.

Historical memories are preset to contain only 0.5 values for both, scaling factor and crossover rate parameters Eq. (**2**).

$$M_{CR,i} = M_{F,i} = 0.5 \text{ for } i = 1, \ldots, H, \tag{2}$$

where H is a user-defined size of historical memories.

Also, the external archive of inferior solutions **A** has to be initialized. Because of no previous inferior solutions, it is initialized empty, **A** = Ø. And index k for historical memory updates is initialized to 1.

The following steps are repeated over the generations until the stopping criterion is met.

Mutation

Mutation strategy "current-to-pbest/1" was introduced in [**14**] and it combines four mutually different vectors in creation of the mutated vector **v**. Therefore, $\mathbf{x}_{pbest} = \mathbf{X}_{r1} \neq \mathbf{X}_{r2} \neq \mathbf{X}_i$ **Eq**. (**3**).

$$\boldsymbol{v}_i = \boldsymbol{x}_i + F_i(\boldsymbol{x}_{pbest} - \boldsymbol{x}_i) + F_i(\boldsymbol{x}_{r1} - \boldsymbol{x}_{r2}), \tag{3}$$

where $\mathbf{x}_{pbest}$ is randomly selected individual from the best NP x p individuals in the current population. The p value is randomly generated for each mutation by PRNG with uniform distribution from the range [p_{min}, 0.2] and p_{min} = 2/NP. Vector $\mathbf{x}_{r1}$ is randomly selected from the current population **P**. Vector $\mathbf{x}_{r2}$ is randomly selected from the union of the current population **P** and external archive A. The scaling factor value F_i is given by Eq. (**4**).

$$F_i = C\left[M_{F,r}, 0.1\right], \tag{4}$$

where M_{Fr} is a randomly selected value (index r is generated by PRNG from the range 1 to H) from $\mathbf{M}_F$ memory and C stands for Cauchy distribution. Therefore the F_i value is generated from the Cauchy distribution with location parameter value M_{Fr} and scale parameter value of 0.1. If the generated value F, higher than 1, it is truncated to 1 and if it is F_i less or equal to 0, it is generated again by Eq. (**4**).

Crossover

In the crossover step, trial vector u is created from the mutated v and original x vectors. For each vector component, a PRNG with uniform distribution is used to generate a random value. If this random value is less or equal to given crossover rate value CR,, current vector component will be taken from a trial vector. Otherwise, it will be taken from the original vector Eq. (**5**). There is also a safety measure, which ensures, that at least one vector component will be taken from the trial vector. This is given by a randomly generated component index j_{rand}.

$$u_{j,i} = \begin{cases} v_{j,i} & \text{if } U[0,1] \leq CR_i \text{ or } j = j_{rand} \\ x_{j,i} & \text{otherwise} \end{cases} . \tag{5}$$

The crossover rate value CR, is generated from a Gaussian distribution with a mean parameter value $\mathbf{M}_{CRr}$ selected from the crossover rate historical memory MCR by the same index r as in the scaling factor case and standard deviation value of 0.1 Eq. (**6**).

$$CR_i = N\left[M_{CR,r}, 0.1\right]. \tag{6}$$

When the generated CR, value is less than 0, it is replaced by 0 and when it is greater than 1, it is replaced by 1.

Selection

The selection step ensures that the optimization will progress towards better solutions because it allows only individuals of better or at least equal objective function value to proceed into the next generation G + 1 Eq. (**7**).

$$\boldsymbol{x}_{i,G+1} = \begin{cases} \boldsymbol{u}_{i,G} & \text{if } f\left(\boldsymbol{u}_{i,G}\right) \leq f\left(\boldsymbol{x}_{i,G}\right) \\ \boldsymbol{x}_{i,G} & \text{otherwise} \end{cases} , \tag{7}$$

where G is the index of the current generation.

Historical Memory Updates

Historical memories $\mathbf{M}_F$ and $\mathbf{M}_{CR}$ are initialized according to Eq. (**2**), but their components change during the evolution. These memories serve to hold successful values of F and CR used in mutation and crossover steps. Successful regarding producing trial individual better than the original individual. During every single generation, these successful values are stored in their corresponding arrays $\mathbf{S}_F$ and $\mathbf{S}_{CR}$. After each generation, one cell of $\mathbf{M}_F$ and $\mathbf{M}_{CR}$ memories is updated. This cell is given by the index k, which starts at 1 and increases by 1 after each generation. When it overflows the memory size H, it is reset to 1. The new value of k-th cell for $\mathbf{M}_F$ is calculated by Eq. (**8**) and for $\mathbf{M}_{CR}$ by Eq. (**9**).

$$M_{F,k} = \begin{cases} \text{mean}_{WL}(\boldsymbol{S}_F) & \text{if } \boldsymbol{S}_F \neq \varnothing \\ M_{F,k} & \text{otherwise} \end{cases}, \tag{8}$$

$$M_{CR,k} = \begin{cases} \text{mean}_{WL}(\boldsymbol{S}_{CR}) & \text{if } \boldsymbol{S}_{CR} \neq \varnothing \\ M_{CR,k} & \text{otherwise} \end{cases}, \tag{9}$$

where $\text{mean}_{WL}()$ stands for weighted Lehmer mean Eq. (**10**).

$$\text{mean}_{WL}(S) = \frac{\sum_{k=1}^{|S|} w_k \cdot S_k^2}{\sum_{k=1}^{|S|} w_k \cdot S_k}, \tag{10}$$

where the weight vector w is given by Eq. (**11**) and is based on the improvement in objective function value between trial and original individuals in current generation G.

$$w_k = \frac{\text{abs}\left(f(\boldsymbol{u}_{k,G}) - f(\boldsymbol{x}_{k,G})\right)}{\sum_{m=1}^{|S_{CR}|} \text{abs}\left(f(\boldsymbol{u}_{m,G}) - f(\boldsymbol{x}_{m,G})\right)}. \tag{11}$$

And since both arrays SF and SCR have the same size, it is arbitrary which size will be used for the upper boundary for m in Eq. (**11**).

2.2 liteSHADE

The lightweight SHADE variant is based on the previous experiments with the SHADE algorithm and represents a direct approach to some of the unnecessary steps in the original SHADE design. For example, the external archive in the mutation step is not used, and the reason for that came from [**15**], where it was shown, that there is no direct impact to the performance of the algorithm when the archive is not used. Also, an updated version of the distance based parameter adaptation [**13**] is used for the update of memory values of F and CR. The list of all changes to the SHADE algorithm is following:

- No archive **A** is used.
- The p in mutation (Eq. (**3**)) is no longer generated randomly, but it is set to 10% of the population size, p = 0.1 * NP.
- The sizes of historical memories of F and CR values ($\mathbf{M}_F$, $\mathbf{M}_{CR}$) are set to 1, H = 1. Therefore, $\mathbf{M}_F$ and $\mathbf{M}_{CR}$ are no longer vectors, but scalars M_F and M_{CR}
- Historical memories store values of F and CR, which moved the individual furthest (Euclidean distance) in the search space during current generation - updated distance based approach independent of the objective function value improvement.
- These memories (M_F, M_{CR}) are initialized to 0.8 instead of 0.5.
- When the Ft and CRt values for mutation and crossover (Eqs. (**4**) and (**6**)) are generated outside of the predefined range, they are generated again to avoid peaks in boundary values (1 for F and 0 and 1 for CR).
-

Algorithm pseudo-code 1: liteSHADE

3

```
1.Set $NP$ and stopping criterion;
2.$G$ = 0, $x_{best}$ = {}, $p_i$ = 0.1*$NP$;
3.Randomly initialize (Eq. (1)) population $\mathbf{P}$ = ($\mathbf{x}_{1,G},\ldots,\mathbf{x}_{NP,G}$);
4.$M_F$ = $M_{CR}$ = 0.8;
5.$\mathbf{P}_{new}$ = {}, $\mathbf{x}_{best}$ = best from population $\mathbf{P}$;
6.while stopping criterion not met do
7.   for $i$ = 1 to $NP$ do
8.      $\mathbf{x}_{i,G}$ = $\mathbf{P}[i]$;
9.      Set $F_i$ by Eq. (4) and $CR_i$ by Eq. (6);
10.        $\mathbf{v}_{i,G}$ by mutation Eq. (3);
11.        $\mathbf{u}_{i,G}$ by crossover Eq. (5);
12.        if $f(\mathbf{u}_{i,G}) < f(\mathbf{x}_{i,G})$ then
13.           $\mathbf{x}_{i,G+1}$ = $\mathbf{u}_{i,G}$;
14.           if distance between $\mathbf{u}_{i,G}$ and $\mathbf{x}_{i,G}$ is the biggest in the
              current generation then $F_i \rightarrow M_F$, $CR_i \rightarrow M_{CR}$;
15.        else
16.           $\mathbf{x}_{i,G+1}$ = $\mathbf{x}_{i,G}$;
17.        end
18.        $\mathbf{x}_{i,G+1} \rightarrow \mathbf{P}_{new}$;
19.     end
20.     $\mathbf{P}$ = $\mathbf{P}_{new}$, $\mathbf{P}_{new}$ = {}, $\mathbf{x}_{best}$ = best from population $\mathbf{P}$, $G$++;
21.  end
22.  return $\mathbf{x}_{best}$ as the best found solution;
```

Experimental Settings

Both algorithms were tested on the CEC 2015 benchmark set of 15 test functions (2 unimodal, 3 simple multimodal, 3 hybrid and 6 composition functions) with accordance to the benchmark requirements. Also the time complexity of both algorithms was measured.

3.1 SHADE and liteSHADE Settings

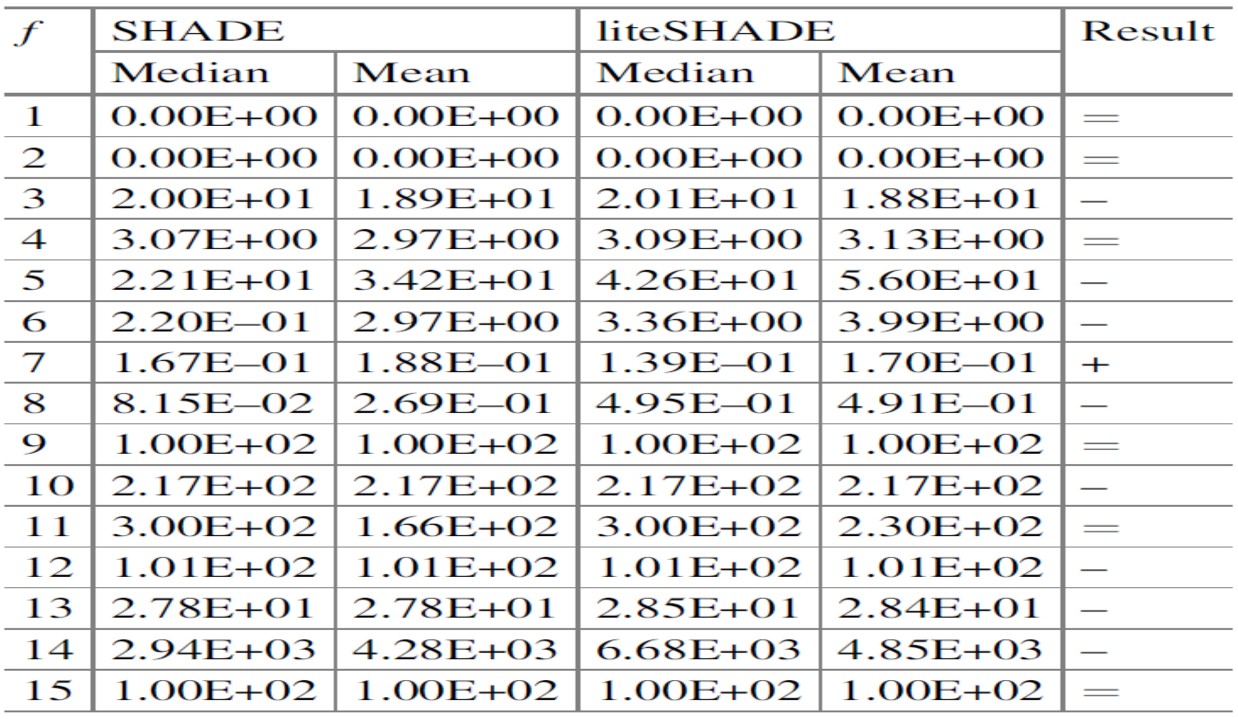
In order to provide the most comparable results, both algorithms had the same setting of control and other parameters:

- Population size NP = 100,
- historical memory size H =10 - SHADE only,
- external archive size |**A**| = NP - SHADE only,
- dimensionality of problems D = {10, 30, 50},
- stopping criterion - maximum number of objective function evaluations MAXFES = 10,000 x D,
- number of runs runs = 51.

4 Results and Discussion

This section provides the results of both algorithms (SHADE and liteSHADE) on the CEC2015 benchmark set in three different dimensional settings - 10D (Table **1**), 30D (Table **2**) and 50D (Table **3**). These tables provide a basic statistical comparison of the median and mean values over the 51 independent runs on each function from the benchmark set and also a result of the Wilcoxon rank-sum test with the significance level set to 5%. When there is no significant difference in the results between both algorithms on a given function, there is an "=" sign in the last column. When the SHADE algorithm performs better, there is a "-" sign and when the liteSHADE performs better, there is a "+" sign.

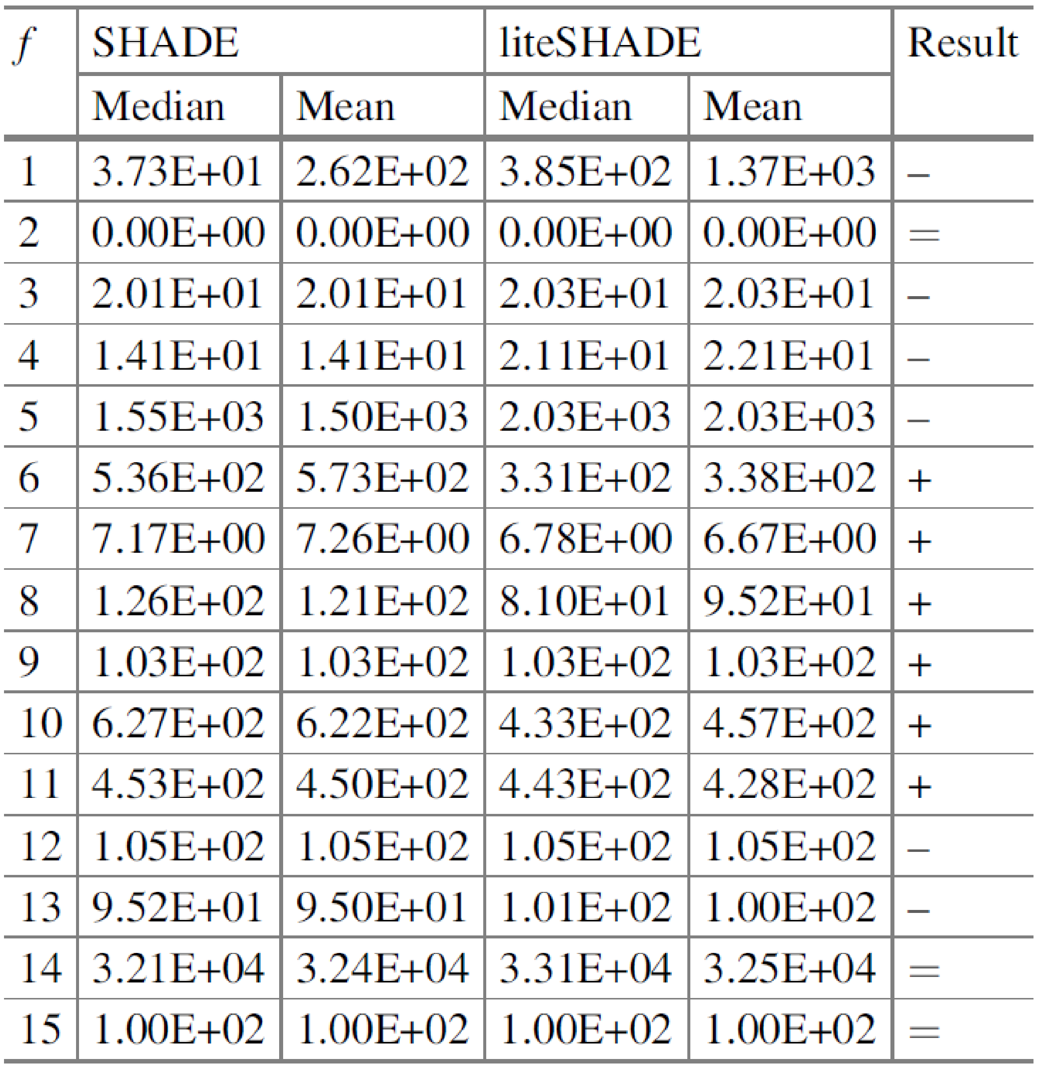
Table 1. SHADE vs. liteSHADE on CEC2015 in 10D.



f	SHADE		liteSHADE		Result
	Median	Mean	Median	Mean	
1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	=
2	0.00E+00	0.00E+00	0.00E+00	0.00E+00	=
3	2.00E+01	1.89E+01	2.01E+01	1.88E+01	−
4	3.07E+00	2.97E+00	3.09E+00	3.13E+00	=
5	2.21E+01	3.42E+01	4.26E+01	5.60E+01	−
6	2.20E−01	2.97E+00	3.36E+00	3.99E+00	−
7	1.67E−01	1.88E−01	1.39E−01	1.70E−01	+
8	8.15E−02	2.69E−01	4.95E−01	4.91E−01	−
9	1.00E+02	1.00E+02	1.00E+02	1.00E+02	=
10	2.17E+02	2.17E+02	2.17E+02	2.17E+02	−
11	3.00E+02	1.66E+02	3.00E+02	2.30E+02	=
12	1.01E+02	1.01E+02	1.01E+02	1.01E+02	−
13	2.78E+01	2.78E+01	2.85E+01	2.84E+01	−
14	2.94E+03	4.28E+03	6.68E+03	4.85E+03	−
15	1.00E+02	1.00E+02	1.00E+02	1.00E+02	=

It can be seen in Table **1**, that in lower dimensional setting, the SHADE algorithm performs better over the whole benchmark set (8 wins, 1 lose and 6 draws), which was predictable, since the simplification of the liteSHADE algorithm provides more explorative power than exploitative. This is confirmed in the case of 30D and 50D problems, where the situation changes and the liteSHADE algorithm can provide better results mostly on the hybrid and composition functions. In 30D the score from the SHADE point of view is 6 wins, 6 loses and 3 draws, and in 50D similarly 6 wins, 5 loses and 4 draws.

Table 2. SHADE vs. liteSHADE on CEC2015 in 30D.



f	SHADE		liteSHADE		Result
	Median	Mean	Median	Mean	
1	3.73E+01	2.62E+02	3.85E+02	1.37E+03	–
2	0.00E+00	0.00E+00	0.00E+00	0.00E+00	=
3	2.01E+01	2.01E+01	2.03E+01	2.03E+01	–
4	1.41E+01	1.41E+01	2.11E+01	2.21E+01	–
5	1.55E+03	1.50E+03	2.03E+03	2.03E+03	–
6	5.36E+02	5.73E+02	3.31E+02	3.38E+02	+
7	7.17E+00	7.26E+00	6.78E+00	6.67E+00	+
8	1.26E+02	1.21E+02	8.10E+01	9.52E+01	+
9	1.03E+02	1.03E+02	1.03E+02	1.03E+02	+
10	6.27E+02	6.22E+02	4.33E+02	4.57E+02	+
11	4.53E+02	4.50E+02	4.43E+02	4.28E+02	+
12	1.05E+02	1.05E+02	1.05E+02	1.05E+02	–
13	9.52E+01	9.50E+01	1.01E+02	1.00E+02	–
14	3.21E+04	3.24E+04	3.31E+04	3.25E+04	=
15	1.00E+02	1.00E+02	1.00E+02	1.00E+02	=

Table 3. SHADE vs. liteSHADE on CEC2015 in 50D.

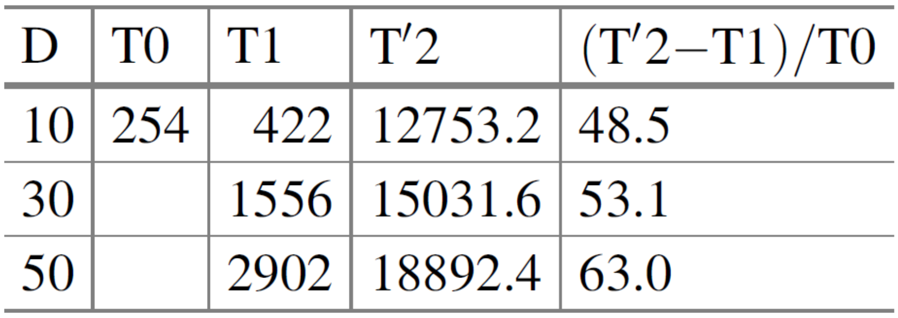
f	SHADE		liteSHADE		Result
	Median	Mean	Median	Mean	
1	1.81E+04	2.14E+04	3.85E+04	4.88E+04	–
2	0.00E+00	0.00E+00	0.00E+00	0.00E+00	=
3	2.01E+01	2.01E+01	2.05E+01	2.05E+01	–
4	3.84E+01	3.92E+01	5.56E+01	5.65E+01	–
5	3.10E+03	3.09E+03	4.38E+03	4.42E+03	–
6	2.87E+03	3.56E+03	2.28E+03	5.90E+03	=
7	4.22E+01	4.25E+01	4.17E+01	4.36E+01	=
8	1.13E+03	1.12E+03	7.35E+02	7.99E+02	+
9	1.06E+02	1.06E+02	1.04E+02	1.04E+02	+
10	1.57E+03	1.59E+03	1.16E+03	1.15E+03	+
11	6.76E+02	6.81E+02	4.77E+02	4.87E+02	+
12	1.08E+02	1.08E+02	1.08E+02	1.08E+02	–
13	1.80E+02	1.80E+02	1.91E+02	1.91E+02	–
14	7.29E+04	6.66E+04	5.92E+04	6.10E+04	+
15	1.00E+02	1.00E+02	1.00E+02	1.00E+02	=

50D problems, where the situation changes and the liteSHADE algorithm can provide better results mostly on the hybrid and composition functions. In 30D the score from the SHADE point of view is 6 wins, 6 loses and 3 draws, and in 50D similarly 6 wins, 5 loses and 4 draws.

These findings support the presumption that the simplified algorithm (liteSHADE) can provide different and competitive results to the original algorithm with higher memory demands (SHADE).

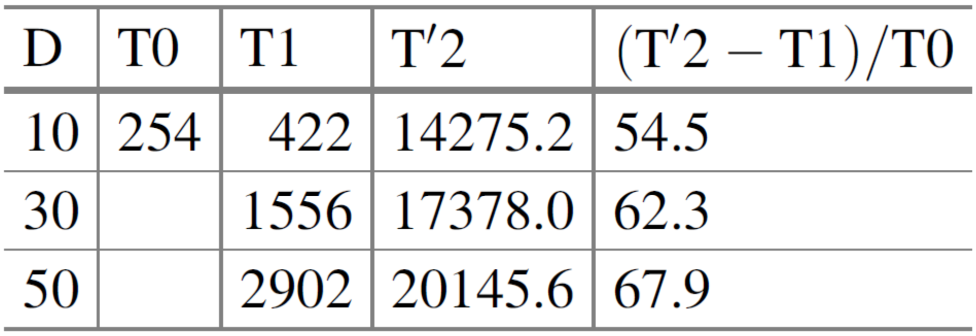
The time complexity measured according to the CEC2015 benchmark set is displayed in Tables 4 and 5, and it can be seen, that the liteSHADE algorithm requires slightly more time to compute. This is most probably caused by the Euclidean distance computation for each individual that improved in the generation. Lowering these time requirements is a subject of future studies in this direction.

Table 4. Time complexity - SHADE.



D	T0	T1	T′2	(T′2−T1)/T0
10	254	422	12753.2	48.5
30		1556	15031.6	53.1
50		2902	18892.4	63.0

Table 5. Time complexity - liteSHADE.



D	T0	T1	T′2	(T′2 − T1)/T0
10	254	422	14275.2	54.5
30		1556	17378.0	62.3
50		2902	20145.6	67.9

5 Conclusion

In this paper, it was shown, that a simplified variant of the SHADE algorithm can provide interesting and competitive results, mostly in higher dimensional settings. The simplified algorithm was coined as liteSHADE. A thorough analysis of its performance and exploration/exploitation abilities will be a next step in the further development.

The goal is to provide a simple, user-friendly, metaheuristic algorithm for global optimization, which would not incorporate complicated mechanisms that introduce new artificial parameters, which should be tuned to the specified problem. The proposed liteSHADE algorithm should be one of the first steps towards reaching this goal.

References

1. Storn, R., Price, K.: Differential Evolution - A Simple and Efficient Adaptive Scheme for Global Optimization Over Continuous Spaces, vol. 3. ICSI, Berkeley (1995)

2. Neri, F., Tirronen, V.: Recent advances in differential evolution: a survey and experimental analysis. Artif. Intell. Rev. 33(1-2), 61-106 (2010)

3. Das, S., Suganthan, P.N.: Differential evolution: a survey of the state-of-the-art. IEEE Trans. Evol. Comput. 15(1), 4-31 (2011)

4. Das, S., Mullick, S.S., Suganthan, P.N.: Recent advances in differential evolution-an updated survey. Swarm Evol. Comput. 27, 1-30 (2016)

5. Gamperle, R., Muller, S.D., Koumoutsakos, P.: A parameter study for differential evolution. Adv. Intell. Syst. Fuzzy Syst. Evol. Comput. 10, 293-298 (2002)

6. Liu, J., Lampinen, J.: On setting the control parameter of the differential evolution method. In: Proceedings of the 8th International Conference on Soft Computing (MENDEL 2002), pp. 11-18 (2002)

7. Tanabe, R., Fukunaga, A.: Success-history based parameter adaptation for differential evolution. In: IEEE Congress on Evolutionary Computation (CEC), pp. 71-78. IEEE, June 2013

8. Tanabe, R., Fukunaga, A.S.: Improving the search performance of SHADE using linear population size reduction. In: IEEE Congress on Evolutionary Computation (CEC), pp. 1658-1665. IEEE, July 2014

9. Guo, S.M., Tsai, J.S.H., Yang, C.C., Hsu, P.H.: A self-optimization approach for L-SHADE incorporated with eigenvector-based crossover and successful-parent-selecting framework on CEC 2015 benchmark set. In: IEEE Congress on Evolutionary Computation (CEC), pp. 1003-1010. IEEE, May 2015

10. Awad, N.H., Ali, M.Z., Suganthan, P.N., Reynolds, R.G.: An ensemble sinusoidal parameter adaptation incorporated with L-SHADE for solving CEC2014 benchmark problems. In: IEEE Congress on Evolutionary Computation (CEC), pp. 2958-2965. IEEE, July 2016

11. Brest, J., Maučec, M.S., Boskovic, B.: Single objective real-parameter optimization: algorithm jSO. In: IEEE Congress on Evolutionary Computation (CEC), pp. 1311-1318. IEEE, June 2017

12. Piotrowski, A.P., Napiorkowski, J.J.: Some metaheuristics should be simplified. Inf. Sci. 427, 32-62 (2018)

13. Viktorin, A., Senkerik, R., Pluhacek, M., Kadavy, T., Zamuda, A.: Distance based parameter adaptation for differential evolution. In: IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1-7. IEEE, November 2017

14. Zhang, J., Sanderson, A.C.: JADE: adaptive differential evolution with optional external archive. IEEE Trans. Evol. Comput. 13(5), 945-958 (2009)

15. Viktorin, A., Senkerik, R., Pluhacek, M., Kadavy, T.: Archive analysis in SHADE. In: International Conference on Artificial Intelligence and Soft Computing, pp. 688-699. Springer, Cham (2017)