*ISSN: 1803-3814*

# CLUSTERING ANALYSIS OF THE POPULATION IN DB_SHADE ALGORITHM

Adam Viktorin, Roman Senkerik, Michal Pluhacek, Tomas Kadavy

Tomas Bata University in Zlin
Faculty of Applied Informatics
T. G. Masaryka 5555, 760 01 Zlin
Czech Republic
{aviktorin, senkerik, pluhacek, kadavy}@utb.cz

Abstract: *This paper provides an analysis of the population clustering in a novel Success-History based Adaptive Differential Evolution algorithm with Distance based adaptation (Db_SHADE) in order to analyze the exploration and exploitation abilities of the algorithm. The comparison with the original SHADE algorithm is performed on the CEC2015 benchmark set in two dimensional settings (10D and 30D). The clustering analysis helps to answer the question about prolonged exploration phase of the Db_SHADE algorithm. Possible future research directions are drawn in the discussion and conclusion.*

Keywords: *Distance based parameter adaptation; SHADE; Differential evolution; DBSCAN.*

## 1 Introduction

The Differential Evolution (DE) algorithm developed by Storn and Price [1] has been in the center of a continuous heuristic optimization since its introduction in 1995. Over the years, many researchers came up with improved versions and the community has grown to an astonishing size. Fortunately, to cover the latest developments in the DE field, three comprehensive surveys were written [2], [3] and [4]. And from these surveys, it can be seen, that the problem of setting control parameters of the DE [5], [6] is usually tackled by a proposal of an adaptive or self-adaptive mechanism. One of the most successful variants with self-adaptive mechanism is Successful-History based Adaptive Differential Evolution (SHADE) [7], which formed a basis for latest winners of the CEC competition in continuous optimization – 2013 SHADE 3rd, 2014 L-SHADE 1st [8], 2015 SPS-L-SHADE-EIG 1st [9], 2016 LSHADE_EpSin 1st [10] and 2017 jSO 1st [11]. Therefore, it was selected for a proposal of a novel Distance based parameter adaptation in [12] and Db_SHADE was developed. Distance based adaptation should promote exploration of the algorithm and avoid premature convergence in higher dimensions.

The hybridization of the DE with other softcomputing fields has shown to be valuable both for analysis and for performance improvement – DE and complex networks [13], [14], [15], DE and chaotic generators [16], [17]. Therefore, in this paper, another softcomputing field, cluster analysis, is used for the analysis of population development in the Db_SHADE algorithm. This combination of two softcomputing fields should answer the question of exploration abilities of the Db_SHADE and provide important insight into the algorithm dynamic.

The rest of the paper is structured as follows: The next section describes DE, SHADE and Db_SHADE algorithms, section 3 depicts the design of experiments, section 4 contains results and their discussion and section 5 ends the whole paper with concluding remarks and possible future research directions.

## 2 DE, SHADE and Db_SHADE

In order to describe the Success-History based Adaptive Differential Evolution algorithm (SHADE) with Distance based parameter adaptation (Db_SHADE), it is important to start from the canonical Differential Evolution (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 $\boldsymbol{P}$, which has a preset size of $NP$. Each individual (solution) in the population consists of a vector $\boldsymbol{x}$ of length $D$ (each vector component corresponds to one attribute of the optimized task) and objective function value $f(\boldsymbol{x})$, which mirrors the quality of the solution. The number of optimized attributes $D$, is often referred to as the dimensionality of the problem. Such generated population $\boldsymbol{P}$ represent the first generation of solutions to the optimization problem.

The individuals in the population are combined in an evolutionary manner in order 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 equal or 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 $\boldsymbol{v}$. This operation uses a control parameter – scaling factor *F*. In the crossover step, a trial vector $\boldsymbol{u}$ is created by selection of attributes either from mutated vector $\boldsymbol{v}$ or the original vector $\boldsymbol{x}$ based on the crossover probability given by a control parameter – crossover rate *CR*. And finally, in the selection, the quality $f(\boldsymbol{u})$ of a trial vector is evaluated by an objective function and compared to the quality $f(\boldsymbol{x})$ 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] and [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. Which 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 along with the novel distance based parameter adaptation.

## 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 [18] and shares a lot of its mechanisms. The main difference is in the historical memories $\boldsymbol{M}_F$ and $\boldsymbol{M}_{CR}$ 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 $\boldsymbol{P}$ is generated randomly and for that matter, a Pseudo-Random Number Generator (PRNG) with uniform distribution is used. Solution vectors $\boldsymbol{x}$ are generated according to the limits of solution space – *lower* and *upper* bounds (1).

$$\boldsymbol{x}_{j,i} = U\left[lower_j, upper_j\right] \text{ for } j = 1, \dots, D; i = 1, \dots, NP \text{ ,} \tag{1}$$

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 (2).

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

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

Also, the external archive of inferior solutions $\boldsymbol{A}$ has to be initialized. Because of no previously inferior solutions, it is initialized empty, $\boldsymbol{A} = \emptyset$. 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-*p*best/1" was introduced in [18] and it combines four mutually different vectors in the creation of the mutated vector $\boldsymbol{v}$. Therefore, $\boldsymbol{x}_{pbest} \neq \boldsymbol{x}_{r1} \neq \boldsymbol{x}_{r2} \neq \boldsymbol{x}_i$ (3).

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

where $\boldsymbol{x}_{pbest}$ is randomly selected individual from the best $NP \times p$ 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 $\boldsymbol{x}_{r1}$ is randomly selected from the current population $\boldsymbol{P}$. Vector $\boldsymbol{x}_{r2}$ is randomly selected from the union of the current population $\boldsymbol{P}$ and external archive $\boldsymbol{A}$. The scaling factor value $F_i$ is given by (4).

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

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

### Crossover

In the crossover step, trial vector $\boldsymbol{u}$ is created from the mutated $\boldsymbol{v}$ and original $\boldsymbol{x}$ 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 $CR_i$, current vector component will be taken from a trial vector. Otherwise, it will be taken from the original vector (5). There is also a safety measure, which ensures, that at least one vector component will be taken from the trial vector. This condition is satisfied by a randomly generated component index $j_{rand}$.

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

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

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

When the generated $CR_i$ 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* (7).

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

where *G* is the index of the current generation.

**Historical Memory Updates**

Historical memories $\boldsymbol{M}_F$ and $\boldsymbol{M}_{CR}$ are initialized according to (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 $\boldsymbol{S}_F$ and $\boldsymbol{S}_{CR}$. After each generation, one cell of $\boldsymbol{M}_F$ and $\boldsymbol{M}_{CR}$ 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 $\boldsymbol{M}_F$ is calculated by (8) and for $\boldsymbol{M}_{CR}$ by (9).

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

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

where mean$_{WL}$() stands for weighted Lehmer (10) mean.

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

where the weight vector $\boldsymbol{w}$ is given by (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 $\boldsymbol{S}_F$ and $\boldsymbol{S}_{CR}$ have the same size, it is arbitrary which size will be used for the upper boundary for *m* in (11).

The last equation (11) is the subject of change in the novel Db_SHADE algorithm, which is described in the next section.

## 2.2 Db_SHADE

The original adaptation mechanism for scaling factor and crossover rate values uses weighted forms of means (10), where weights are based on the improvement in objective function value (11). This approach promotes exploitation over exploration and therefore might lead to premature convergence, which could be a problem especially in higher dimensions.

Distance approach is based on the Euclidean distance between the trial and the original individual, which slightly increases the complexity of the algorithm by exchanging simple difference for Euclidean distance computation for the price of stronger exploration. In this case, scaling factor and crossover rate values connected with the individual that moved the furthest will have the highest weight (12).

$$w_k = \frac{\sqrt{\sum_{j=1}^{D} \left(u_{k,j,G} - x_{k,j,G}\right)^2}}{\sum_{m=1}^{|S_{CR}|} \sqrt{\sum_{j=1}^{D} \left(u_{m,j,G} - x_{m,j,G}\right)^2}}. \tag{12}$$

Therefore, the exploration ability is rewarded and this should lead to avoidance of the premature convergence in higher dimensional objective spaces. Such approach might be also useful for constrained problems, where constrained areas could be overcome by increased changes of individual's components.

Below is the pseudo-code of the Db_SHADE algorithm for a clear overview.

**Algorithm 1: Db_**SHADE

```
1   Set $NP$, $H$ and stopping criterion;
2   $G = 0$, $x_{best} = \{\}$, $k = 1$, $p_{min} = 2/NP$, $\boldsymbol{A} = \varnothing$;
3   Randomly initialize (1) population $\boldsymbol{P} = (\boldsymbol{x}_{1,G},\dots,\boldsymbol{x}_{NP,G})$;
4   Set $\boldsymbol{M}_F$ and $\boldsymbol{M}_{CR}$ according to (2);
5   $\boldsymbol{P}_{new} = \{\}$, $\boldsymbol{x}_{best}$ = best from population $\boldsymbol{P}$;
6   while stopping criterion not met do
7       $\boldsymbol{S}_F = \varnothing$, $\boldsymbol{S}_{CR} = \varnothing$;
8       for $i = 1$ to $NP$ do
9           $\boldsymbol{x}_{i,G} = \boldsymbol{P}[i]$;
10          $r = U[1, H]$, $p_i = U[p_{min}, 0.2]$;
11          Set $F_i$ by (4) and $CR_i$ by (6);
12          $\boldsymbol{v}_{i,G}$ by mutation (3);
13          $\boldsymbol{u}_{i,G}$ by crossover (5);
14          if $f(\boldsymbol{u}_{i,G}) < f(\boldsymbol{x}_{i,G})$ then
15              $\boldsymbol{x}_{i,G+1} = \boldsymbol{u}_{i,G}$; $\boldsymbol{x}_{i,G} \rightarrow \boldsymbol{A}$; $F_i \rightarrow \boldsymbol{S}_F$, $CR_i \rightarrow \boldsymbol{S}_{CR}$;
16          else
17              $\boldsymbol{x}_{i,G+1} = \boldsymbol{x}_{i,G}$;
18          end
19          if $|\boldsymbol{A}| > NP$ then randomly delete $|\boldsymbol{A}| - NP$ individuals from $\boldsymbol{A}$ end;
20          $\boldsymbol{x}_{i,G+1} \rightarrow \boldsymbol{P}_{new}$;
21      end
22      if $\boldsymbol{S}_F \neq \varnothing$ and $\boldsymbol{S}_{CR} \neq \varnothing$ then
23          Update $\boldsymbol{M}_{F,k}$ (8) and $\boldsymbol{M}_{CR,k}$ (9) with distance based weight from (12), $k$++;
24          if $k > H$ then $k = 1$, end
25      end
26      $\boldsymbol{P} = \boldsymbol{P}_{new}$, $\boldsymbol{P}_{new} = \{\}$, $\boldsymbol{x}_{best}$ = best from population $\boldsymbol{P}$;
27  end
28  return $\boldsymbol{x}_{best}$ as the best found solution
```

## 3 Experimental Design

The goal of this paper was to study clustering behavior of the population in Db_SHADE. However, for the purpose of a comparison, the cluster history was recorded for both, original SHADE and Db_SHADE algorithms. This was done for CEC2015 benchmark set of 15 test functions in two dimensional settings – 10*D* and 30*D*. Next two subsections describe the settings for both DE variants and cluster analysis tool.

### 3.1 SHADE and Db_SHADE settings

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

- Population size $NP = 100$,
- historical memory size $H = 10$,
- external archive size $|\boldsymbol{A}| = NP$,
- dimensionality of problems $D = \{10, 30\}$,
- stopping criterion – maximum number of objective function evaluations $MAXFES = 10{,}000 \times D$,
- number of runs $runs = 51$.

### 3.2 Cluster Analysis

The clustering algorithm selected for this experiment is DBSCAN [19], which conveniently works on the basis of cluster density, and therefore can discover clusters of arbitrary shapes.

The DBSCAN algorithm requires setting of two control parameters and a possible distance measure. The settings with an explanation are as follows:

- Core point distance $Eps$ = 1% of the decision space – for CEC2015 benchmark set $Eps = 2$,
- minimal number of points to form a cluster $MinPts = 4$ – minimal number of individuals for mutation,
- distance measure = Chebyshev distance [20] – if the distance between any corresponding attributes of two individuals is higher than 1% of the decision space, they are not considered directly density-reachable.

## 4 Results and Discussion

In this section, the computational results are presented along with the convergence and clustering plots. In order to provide a statistic comparison between SHADE and Db_SHADE algorithms, the results of 51 independent runs were tested by a Wilcoxon rank-sum test with the significance level of 5% on each test function. The outcomes of these tests are provided along the median and mean values in Table 1 for 10*D* problems and in Table 2 for 30*D* problems. Whenever the Db_SHADE algorithm performed significantly better than SHADE, the '+' sign is used in the last column, in case of no significant difference in results, '=' sign is used and if the SHADE performed better, '-' sign would be used. However, this case did not occur.

Table 1: SHADE vs. Db_SHADE on CEC2015 in 10*D*

| **10D** | **SHADE** | | **Db_SHADE** | | |
|---|---|---|---|---|---|
| ***Func*** | ***Median*** | ***Mean*** | ***Median*** | ***Mean*** | ***Result*** |
| *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 | 2.00E+01 | 1.92E+01 | = |
| *4* | 3.07E+00 | 2.97E+00 | 3.06E+00 | 2.98E+00 | = |
| *5* | 2.21E+01 | 3.42E+01 | 2.98E+01 | 4.52E+01 | = |
| *6* | 2.20E-01 | 2.97E+00 | 4.16E-01 | 8.08E-01 | = |
| *7* | 1.67E-01 | 1.88E-01 | 1.73E-01 | 1.91E-01 | = |
| *8* | 8.15E-02 | 2.69E-01 | 4.28E-02 | 2.06E-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 | 2.01E+02 | = |
| *12* | 1.01E+02 | 1.01E+02 | 1.01E+02 | 1.01E+02 | + |
| *13* | 2.78E+01 | 2.78E+01 | 2.79E+01 | 2.76E+01 | = |
| *14* | 2.94E+03 | 4.28E+03 | 2.98E+03 | 4.66E+03 | = |
| *15* | 1.00E+02 | 1.00E+02 | 1.00E+02 | 1.00E+02 | = |

Table 2: SHADE vs. Db_SHADE on CEC2015 in 30*D*

| **10D** | **SHADE** | | **Db_SHADE** | | |
|---|---|---|---|---|---|
| ***Func*** | ***Median*** | ***Mean*** | ***Median*** | ***Mean*** | ***Result*** |
| *1* | 3.73E+01 | 2.62E+02 | 2.12E+01 | 2.42E+02 | = |
| *2* | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | = |
| *3* | 2.01E+01 | 2.01E+01 | 2.01E+01 | 2.01E+01 | = |
| *4* | 1.41E+01 | 1.41E+01 | 1.32E+01 | 1.31E+01 | = |
| *5* | 1.55E+03 | 1.50E+03 | 1.54E+03 | 1.52E+03 | = |
| *6* | 5.36E+02 | 5.73E+02 | 3.37E+02 | 3.48E+02 | + |
| *7* | 7.17E+00 | 7.26E+00 | 6.81E+00 | 6.74E+00 | + |
| *8* | 1.26E+02 | 1.21E+02 | 5.27E+01 | 7.38E+01 | + |
| *9* | 1.03E+02 | 1.03E+02 | 1.03E+02 | 1.03E+02 | + |
| *10* | 6.27E+02 | 6.22E+02 | 5.29E+02 | 5.32E+02 | + |
| *11* | 4.53E+02 | 4.50E+02 | 4.10E+02 | 4.16E+02 | + |
| *12* | 1.05E+02 | 1.05E+02 | 1.05E+02 | 1.05E+02 | = |
| *13* | 9.52E+01 | 9.50E+01 | 9.47E+01 | 9.50E+01 | = |
| *14* | 3.21E+04 | 3.24E+04 | 3.22E+04 | 3.24E+04 | = |
| *15* | 1.00E+02 | 1.00E+02 | 1.00E+02 | 1.00E+02 | = |

It can be seen, that the performance in 10*D* is quite comparable and there is only one case – *f*12, where the Db_SHADE algorithm performs significantly better. This is a predicted situation because the distance based parameter adaptation targets premature convergence, which is an issue in higher dimensional decision spaces.

The situation is more interesting in the case of 30-dimensional decision space, where there are 6 significant wins for the Db_SHADE algorithm on multimodal and hybrid functions – *f*6 to *f*11. These functions were also selected for the depiction of average convergence and average cluster count development in Figures 1, 2 and 3. The cluster development plots also show a 95% confidence interval in lighter colors.

As can be seen from figures, clusters in population start to appear later in the case of the Db_SHADE algorithm on functions *f*6, *f*8, *f*10 and *f*11, which supports the claim of slower convergence of the whole population and better exploration abilities at the start of the optimization process.

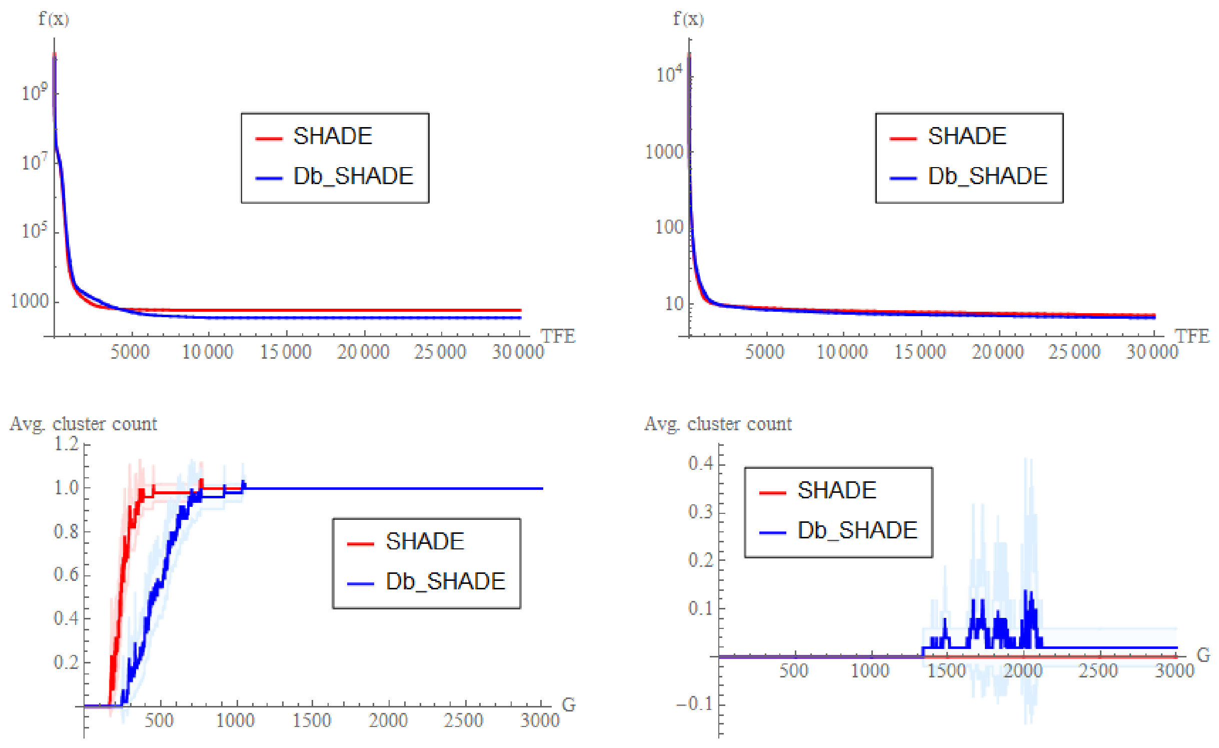

Figure 1: Convergence plots (top) and cluster development plots (bottom) of CEC2015 test functions *f*6 (left) and *f*7 (right) in 30*D*

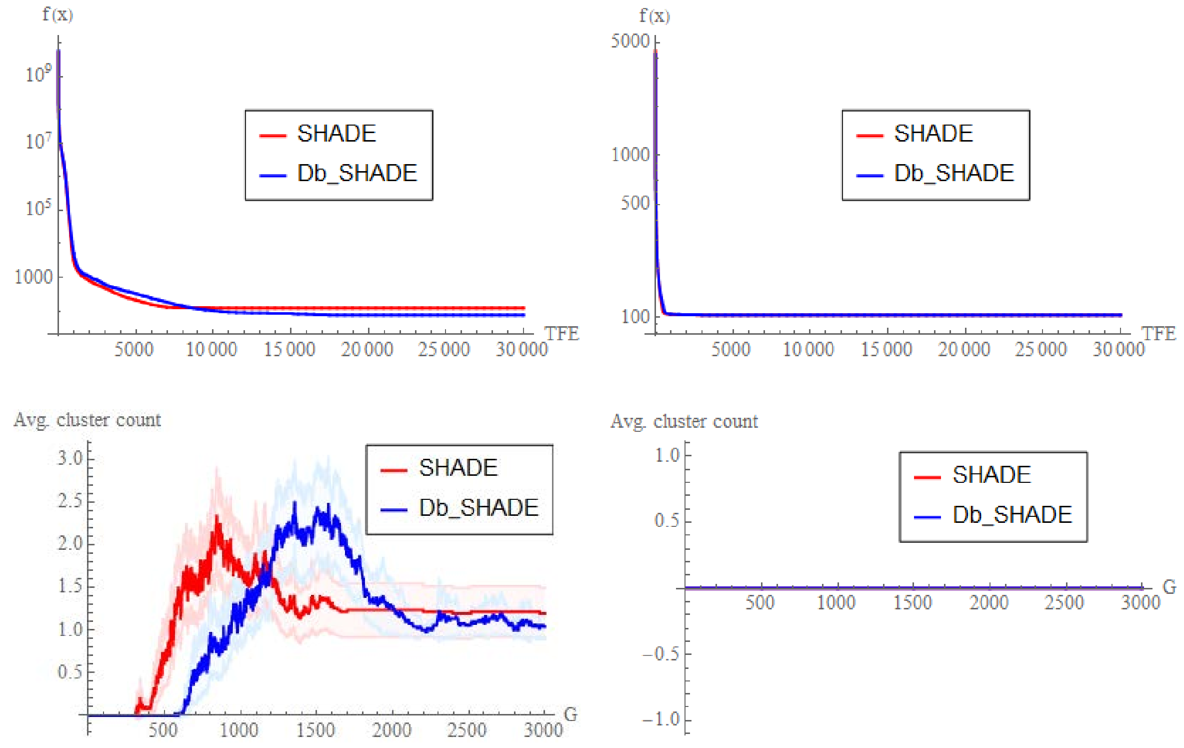

Figure 2: Convergence plots (top) and cluster development plots (bottom) of CEC2015 test functions *f*8 (left) and *f*9 (right) in 30*D*

In the case of *f*9, neither of algorithms form clusters at all, which might be caused by easily reachable decision space area with the same objective function value, where the algorithm gets stuck.

And an interesting development can be seen in the case of *f*7, where Db_SHADE algorithm forms clusters in contrast with the original SHADE, which apparently does not. This is a promising area for future analysis.

Overall, from the point of view of convergence, it can be seen, that when the population starts forming clusters, algorithm quickly converges to local optima and is unable to escape it and explore further. This behavior is very similar to swarm based algorithms. However, it creates an interesting direction for the future research of population control mechanisms based on the cluster analysis, which is a planned extension of this work.

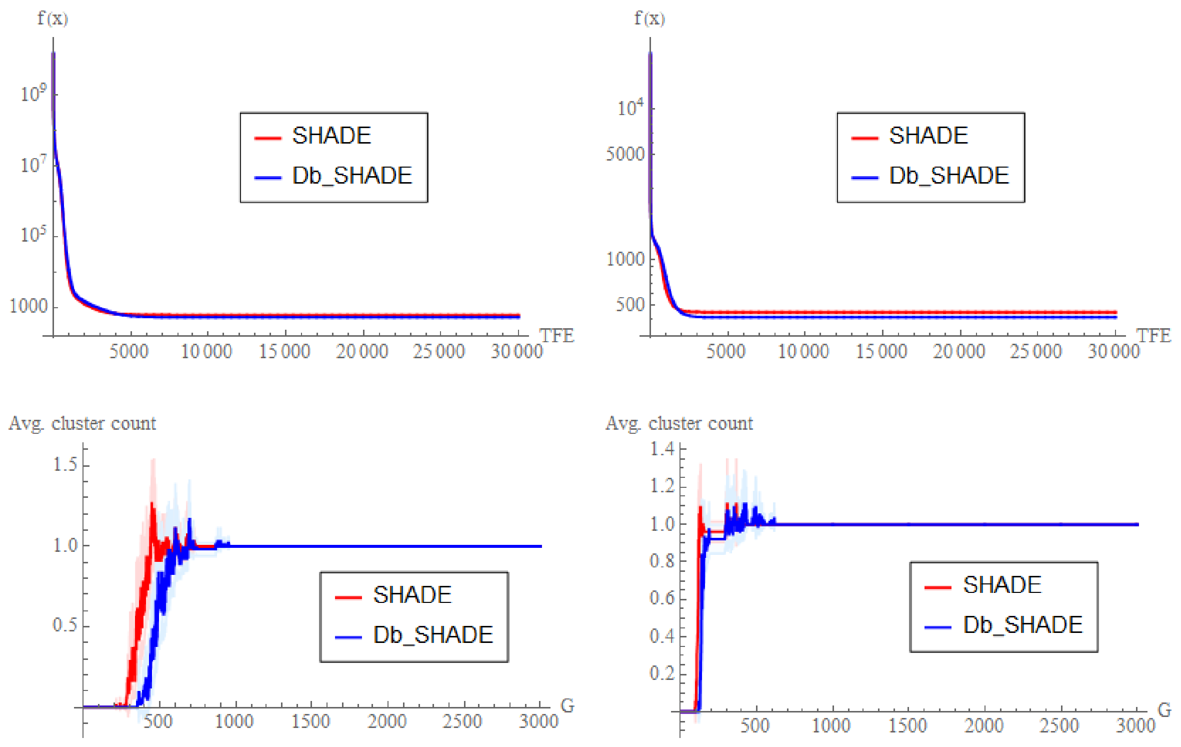

Figure 3: Convergence plots (top) and cluster development plots (bottom) of CEC2015 test functions *f*10 (left) and *f*11 (right) in 30*D*

## 5 Conclusion

In this paper, a combination of two softcomputing fields was presented – evolutionary computation and cluster analysis. In order to characterize the behavior of a population in SHADE and Db_SHADE algorithms, a DBSCAN clustering was utilized and the analysis was performed on the CEC2015 benchmark set.

The results confirm the hypothesis about longer exploration abilities of the distance based DE version and provide interesting possible directions for future research. In comparison with the popular linear decrease of population size, the cluster analysis results might be used to produce a more sophisticated approach to population size management.

**Acknowledgement:** This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic within the National Sustainability Programme Project no. LO1303 (MSMT-7778/2014), further by the European Regional Development Fund under the Project CEBIA-Tech no. CZ.1.05/2.1.00/03.0089 and by Internal Grant Agency of Tomas Bata University under the Projects no. IGA/CebiaTech/2018/003. This work is also based upon support by COST (European Cooperation in Science & Technology) under Action CA15140, Improving Applicability of Nature-Inspired Optimisation by Joining Theory and Practice (ImAppNIO), and Action IC1406, High-Performance Modelling and Simulation for Big Data Applications (cHiPSet). The work was further supported by resources of A.I.Lab at the Faculty of Applied Informatics, Tomas Bata University in Zlin (ailab.fai.utb.cz).

## 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, Artificial Intelligence Review, **33**(1-2), 61-106 (2010)

[3] Das, S., Suganthan, P. N.: Differential evolution: a survey of the state-of-the-art. IEEE transactions on evolutionary computation, **15**(1), 4-31 (2011)

[4] Das, S., Mullick, S. S., Suganthan, P. N.: Recent advances in differential evolution–An updated survey. Swarm and Evolutionary Computation, **27**, 1-30 (2016)

[5] Gämperle, R., Müller, S. D., Koumoutsakos, P.: A parameter study for differential evolution. Advances in intelligent systems, fuzzy systems, evolutionary computation, **10**, 293-298 (2002)

[6] Liu, J., Lampinen, J.: On setting the control parameter of the differential evolution method. In: Proceedings of 8th International Conference on Soft Computing – MENDEL 2002, no. 8 in MENDEL, pp. 11-18. Brno University of Technology, VUT press, Brno (2002)

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

[8] Tanabe, R., Fukunaga, A. S.: Improving the search performance of SHADE using linear population size reduction. In: Evolutionary Computation (CEC), 2014 IEEE Congress on, pp. 1658-1665. IEEE (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: Evolutionary Computation (CEC), 2015 IEEE Congress on, pp. 1003-1010. IEEE (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: Evolutionary Computation (CEC), 2016 IEEE Congress on, pp. 2958-2965. IEEE (2016)

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

[12] Viktorin, A., Senkerik, R., Pluhacek, M., Kadavy, T. Zamuda, A.: Distance Based Parameter Adaptation for Differential Evolution. In: Computational Intelligence (SSCI), 2017 IEEE Symposium Series on, pp. 1-7 IEEE (2017)

[13] Viktorin, A., Senkerik, R., Pluhacek, M., Kadavy, T.: SHADE Algorithm Dynamic Analyzed Through Complex Network. In: International Computing and Combinatorics Conference, pp. 666-677. Springer, Cham (2017)

[14] Viktorin, A., Pluhacek, M., Senkerik, R., Kadavy, T.: Detecting Potential Design Weaknesses in SHADE Through Network Feature Analysis. In: International Conference on Hybrid Artificial Intelligence Systems, pp. 662-673. Springer, Cham (2017)

[15] Senkerik, R., Viktorin, A., Pluhacek, M.: On the transforming of the indices selection mechanism inside differential evolution into complex network. In: Intelligent Networking and Collaborative Systems (INCoS), 2016 International Conference on, pp. 186-192. IEEE (2016)

[16] Viktorin, A., Pluhacek, M., Senkerik, R.: Success-history based adaptive differential evolution algorithm with multi-chaotic framework for parent selection performance on CEC2014 benchmark set. In: Evolutionary Computation (CEC), 2016 IEEE Congress on, pp. 4797-4803. IEEE (2016)

[17] Senkerik, R., Pluhacek, M., Viktorin, A., Kadavy, T.: On the Randomization of Indices Selection for Differential Evolution. In: Computer Science On-line Conference, pp. 537-547. Springer, Cham (2017)

[18] Zhang, J., Sanderson, A. C.: JADE: adaptive differential evolution with optional external archive. Evolutionary Computation, IEEE Transactions on, **13**(5), 945-958 (2009)

[19] Ester, M., Kriegel, H. P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In Kdd, **96**(34), 226-231 (1996)

[20] Deza, M. M., Deza, E.: Encyclopedia of distances. In: Encyclopedia of Distances. Springer, Berlin, Heidelberg (2009)