ANALYSING KNOWLEDGE TRANSFER IN SHADE VIA COMPLEX NETWORK

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}@fai.utb.cz

Abstract: *In this research paper a hybridization of two computational intelligence fields, which are evolutionary computation techniques and complex networks, is presented. During the optimization run of the Success-History based Adaptive Differential Evolution (SHADE) a complex network is built and its feature, node degree centrality, is extracted for each node. Nodes represent here the individual solutions from the SHADE population. Edges in the network mirror the knowledge transfer between individuals in SHADE's population, and therefore, the node degree centrality can be used to measure knowledge transfer capabilities of each individual. The correlation between individual's quality and its knowledge transfer capability is recorded and analyzed on the CEC2015 benchmark set in three different dimensionality settings – 10D, 30D, and 50D. Results of the analysis are discussed, and possible directions for future research are suggested.*

Keywords: *Differential Evolution; SHADE; Complex Network; Centrality; Knowledge Transfer*

1 Introduction

The Differential Evolution (DE) algorithm was proposed by Storn and Price in 1995 [1] to solve optimization problems in the non-linear continuous domain. Since then, it is one of the best performing optimization algorithms from the family of Evolutionary Computational Techniques (ECTs) with a notable scientific community [2]–[4]. Every year, the community produces a large number of improvements, updates and even new versions of the original DE algorithm, which aim to improve the performance and robustness of the DE. This leads to a need for an understanding of the problem domain, to select appropriate DE variant, which might be a problem for a non-experienced user. Luckily, over the last few years, there have been published surveys to help distinguish between numerous DE variants [5]–[7].

There is another critical aspect of using DE for optimization tasks, and that is an appropriate setting of its control parameters. The canonical DE from 1995 incorporates three control parameters (apart from the max. number of generations) – population size *NP*, scaling factor *F* and crossover rate *CR*. Algorithms performance is heavily influenced by these three parameters [8], [9], and therefore, there has been an endeavor for a robust parameter-less DE, which would serve users, without a need for expert knowledge in the DE field. The late trend in this research direction is an adaptation of the control parameters to a given task during the optimization itself. In a sense, these algorithms try to overcome the famous No Free Lunch (NFL) theorem [10]. Typical examples are SDE [11] with self-adaptive scaling factor and crossover rate generated from normal distribution, jDE [12] with scaling factor and crossover rate values encoded into individuals, MDE-*p*BX [13] with adaptive scaling factor generated from Cauchy distribution, adaptive crossover rate generated from Gaussian distribution and a new mutation scheme "current-to-gr_best/1", SaDE [14] with self-adaptive mutation strategies along with scaling factor and crossover rate, JADE [15] with a novel mutation strategy "current-to-*p*best/1" and adaptation of both scaling factor and crossover rate, SHADE [16] which is an improved variant of JADE with historical memories of successful control parameters and L-SHADE [17] which adds linear decrease of population size (simple population management) into the SHADE algorithm.

The performance abilities of adaptive DE variants [18]–[26] are notable from recent competitions in numerical optimization, where especially variants of Success-History based Adaptive Differential Evolution (SHADE) were successful: 3rd place on CEC2013 – SHADE [16], 1st place on CEC2014 – L-SHADE [17], 1st place on CEC2015 – SPS-L-SHADE-EIG [27], joint 1st place on CEC2016 – LSHADE_EpSin [28] and 1st place on CEC2017 – jSO [29].

Adaptive DE variants were also successfully used in numerous real-world applications, i.e., for solving the vehicle routing problem with profits [30], automatic test case generation [31], underwater glider path planning [32], high-rise building design [33], power system stabilizer design [34] and in designing of optimal harmonic filters [35].

Thanks to the rising popularity of SHADE algorithm variants in both benchmark and real problem domains, there is a need for a thorough analysis of its inner dynamics, capabilities and design weaknesses, because there are still challenges in the understanding of the control parameter properties [36] and their online effects [37]. Moreover, according to the recent study by Tanabe and Fukunaga [38], there is still much room-for-improvement in the adaptive algorithm design.

Therefore, this paper presents a possible future direction in the analysis and design of the SHADE algorithm by combining the field of ECTs with the field of Complex Networks (CNs).
Currently, the utilization of CN as a visualization tool for the analysis of population dynamics for evolutionary and swarm-based algorithms is becoming an attractive open research task. The population is visualized as an evolving complex network that exhibits non-trivial features. These features offer a clear description of the population under evaluation and can be utilized for the adaptive population as well as parameter control during the metaheuristic run. The initial studies [39]–[41] describing the possibilities of transforming population dynamics into complex networks, were followed by the successful adaptation and control of the metaheuristic algorithm during the run through the given complex networks' frameworks [42]–[44].
The idea of a mutual connection between ECTs and CN was already expanded to the SHADE algorithm universum in [45] and with the feedback to the running algorithm in [46], where the inner dynamics of the SHADE was translated into the CN and its features were used as guidance for linear population size decrease. This paper builds on these foundations and proposes an analysis of the knowledge transfer between individuals of SHADE algorithm in each generation by analyzing the node degree centrality of CN created during mutation, crossover and selection steps and its correlation to the quality of individual solutions.
The remainder of this paper is structured as follows: Sections 2 and 3 are dedicated to DE and SHADE algorithms, Section 4 describes the network design and scoring methods for individuals. Section 5 depicts experimental setting, Section 6 discusses results, and the last section of this paper contains the concluding remarks.

2 Differential Evolution

The DE algorithm is initialized with a random population of individuals $\boldsymbol{P}$, that represent solutions to the optimization problem. The population size NP is set by the user along with other control parameters – scaling factor F and crossover rate CR.
In continuous optimization, each individual is composed of a vector $\boldsymbol{x}$ of length D, which is a dimensionality (number of optimized attributes) of the problem, and each vector component represents a value of the corresponding attribute, and of objective function value $f(\boldsymbol{x})$.
For each individual in a population, three mutually different individuals are selected for mutation of vectors and resulting mutated vector $\boldsymbol{v}$ is combined with the original vector $\boldsymbol{x}$ in crossover step. The objective function value $f(\boldsymbol{u})$ of the resulting trial vector $\boldsymbol{u}$ is evaluated and compared to that of the original individual. When the quality (objective function value) of the trial individual is better, it is placed into the next generation. Otherwise, the original individual is placed there. This step is called selection. The process is repeated until the stopping criterion is met (e.g., the maximum number of objective function evaluations, the maximum number of generations, the low bound for diversity between objective function values in population).
The following sections describe four steps of DE: Initialization, mutation, crossover, and selection.

2.1 Initialization

As aforementioned, the initial population $\boldsymbol{P}$, of size NP, of individuals is randomly generated. For this purpose, the individual vector $\boldsymbol{x}_i$ components are generated by Random Number Generator (RNG) with uniform distribution from the range which is specified for the problem by *lower* and *upper* bound (1).

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

where i is the index of a current individual, j is the index of current attribute and D is the dimensionality of the problem.
In the initialization phase, a scaling factor value F and crossover value CR has to be assigned as well. The typical range for F value is [0, 2] and for CR, it is [0, 1].

2.2 Mutation

In the mutation step, three mutually different individuals $\boldsymbol{x}_{r1}$, $\boldsymbol{x}_{r2}$, $\boldsymbol{x}_{r3}$ from a population are randomly selected and combined in mutation according to the mutation strategy. The original mutation strategy of canonical DE is "rand/1" and is depicted in (2).

$$\boldsymbol{v}_i = \boldsymbol{x}_{r1} + F(\boldsymbol{x}_{r2} - \boldsymbol{x}_{r3}), \tag{2}$$

where $r1 \neq r2 \neq r3 \neq i$, F is the scaling factor value, and $\boldsymbol{v}_i$ is the resulting mutated vector.

2.3 Crossover

In the crossover step, mutated vector $\boldsymbol{v}_i$ is combined with the original vector $\boldsymbol{x}_i$ and produces trial vector $\boldsymbol{u}_i$. The binary crossover (3) is used in canonical DE.

$$u_{j,i} = \begin{cases} v_{j,i} & \text{if } U[0,1] \leq CR \text{ or } j = j_{rand} \\ x_{j,i} & \text{otherwise} \end{cases}, \tag{3}$$

where CR is the used crossover rate value, and j_{rand} is an index of an attribute that has to be from the mutated vector $\boldsymbol{v}_i$ (ensures generation of a vector with at least one new component).

2.4 Selection

The selection step ensures that the optimization progress will lead to better solutions because it allows only individuals of better or at least equal objective function value to proceed into the next generation *G+1* (4).

$$\boldsymbol{x}_{i,G+1} = \begin{cases} \boldsymbol{u}_{i,G} & \text{if } f(\boldsymbol{u}_{i,G}) \leq f(\boldsymbol{x}_{i,G}) \\ \boldsymbol{x}_{i,G} & \text{otherwise} \end{cases}, \tag{4}$$

where *G* is the index of the current generation.
The whole DE algorithm is depicted in pseudo-code below.

```
Algorithm pseudo-code 1: DE
Set NP, CR, F and stopping criterion;
G = 0, x_best = {};
Randomly initialize (1) population P = (x_1,G, …, x_NP,G);
P_new = {}, x_best = best from population P;
while stopping criterion not met
  for i = 1 to NP do
    x_i,G = P[i];
    v_i,G by mutation (2);
    u_i,G by crossover (3);
    if f(u_i,G) < f(x_i,G) then
      x_i,G+1 = u_i,G;
    else
      x_i,G+1 = x_i,G;
    end
    x_i,G+1 → P_new;
  end
  P = P_new, P_new = {}, x_best = best from population P;
end
return x_best as the best found solution
```

3 Success-History based Adaptive Differential Evolution

In SHADE, the only control parameter that can be set by the user is population size *NP*, other two parameters (*F*, *CR*) are adapted to the given optimization task. A new parameter *H* is introduced, which determines the sizes of *F* and *CR* value memories. The initialization step of the SHADE is, therefore, similar to DE. Mutation, however, is entirely different because of the used strategy "current-to-*p*best/1" and the fact, that it uses different scaling factor value F_i for each individual. Mutation strategy also works with a new feature – external archive of inferior solutions. This archive holds individuals from previous generations, which were outperformed in the selection step. The size of the archive retains the same size as the size of the population by randomly discarding its contents whenever the size overflows *NP*.
Crossover is still binary, but similarly to the mutation and scaling factor values, crossover rate value CR_i is also different for each individual.
The selection step is the same and therefore following sections describe only different aspects of initialization, mutation, and crossover.

3.1 Initialization

As aforementioned, initial population $\boldsymbol{P}$ is randomly generated as in DE, but additional memories for *F* and *CR* values are initialized as well. Both memories have the same size *H* and are equally initialized, the memory for *CR* values is titled $\boldsymbol{M}_{CR}$ and the memory for *F* is titled $\boldsymbol{M}_F$. Their initialization is depicted in (5).

$$M_{CR,i} = M_{F,i} = 0.5 \text{ for } i = 1, \dots, H. \tag{5}$$

Also, the external archive of inferior solutions $\boldsymbol{A}$ is initialized. Since there are no solutions so far, it is initialized empty $\boldsymbol{A} = \emptyset$ and its maximum size is set to *NP*.

3.2 Mutation

Mutation strategy "current-to-*p*best/1" was introduced in [15] and unlike "rand/1", it combines four mutually different vectors, therefore $pbest \neq r1 \neq r2 \neq i$ (6).

$$\boldsymbol{v}_i = \boldsymbol{x}_i + F_i(\boldsymbol{x}_{pbest} - \boldsymbol{x}_i) + F_i(\boldsymbol{x}_{r1} - \boldsymbol{x}_{r2}), \tag{6}$$

where $\boldsymbol{x}_{pbest}$ is randomly selected from the best $NP \times p$ best individuals in the current population. The *p* value is randomly generated for each mutation by RNG with uniform distribution from the range $[p_{min}, 0.2]$. where $p_{min} = 2/NP$. Vector $\boldsymbol{x}_{r1}$

is randomly selected from the current population, and vector $\boldsymbol{x}_{r2}$ is randomly selected from the union of current population $\boldsymbol{P}$ and archive $\boldsymbol{A}$. The scaling factor value F_i is given by (7).

$$F_i = C\left[M_{F,r}, 0.1\right], \tag{7}$$

where $M_{F,r}$ is a randomly selected value (by index r) 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 0.1. If the generated value $F_i > 1$, it is truncated to 1 and if it is $F_i \leq 0$, it is generated again by (7).

3.3 Crossover

Crossover is the same as in (3), but the CR value is changed to CR_i, which is generated separately for each individual (8). The value is generated from the Gaussian distribution with a mean parameter value of $M_{CR,r}$, which is randomly selected (by the same index r as in mutation) from $\boldsymbol{M}_{CR}$ memory and standard deviation value of 0.1.

$$CR_i = N\left[M_{CR,r}, 0.1\right]. \tag{8}$$

3.4 Historical Memory Updates

Historical memories $\boldsymbol{M}_F$ and $\boldsymbol{M}_{CR}$ are initialized according to (5), but its components change during the evolution. These memories serve to hold successful values of F and CR used in mutation and crossover steps. Successful in terms of producing trial individual better than the original individual. During one generation, these successful values are stored in 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 size limit of memories H, it is again set to 1. The new value of k-th cell for $\boldsymbol{M}_F$ is calculated by (9) and for $\boldsymbol{M}_{CR}$ by (10).

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

$$M_{CR,k} = \begin{cases} \text{mean}_{WL}(\boldsymbol{S}_{CR}) & \text{if } \boldsymbol{S}_{CR} \neq \emptyset \\ M_{CR,k} & \text{otherwise'} \end{cases} \tag{10}$$

where $\text{mean}_{WL}()$ stands for weighted Lehmer mean (11).

$$\text{mean}_{WL}(\boldsymbol{S}) = \frac{\sum_{k=1}^{|S|} w_k \cdot S_k^2}{\sum_{k=1}^{|S|} w_k \cdot S_k}, \tag{11}$$

where the weight vector $\boldsymbol{w}$ is given by (12) and is based on the improvement in objective function value between trial and original individuals.

$$w_k = \frac{\text{abs}\left(f(\boldsymbol{u}_{k,G}) - f(\boldsymbol{x}_{k,G})\right)}{\sum_{m=1}^{|S|} \text{abs}\left(f(\boldsymbol{u}_{m,G}) - f(\boldsymbol{x}_{m,G})\right)}. \tag{12}$$

Moreover, since both arrays $\boldsymbol{S}_F$ and $\boldsymbol{S}_{CR}$ have the same size, it is arbitrary which size will be used as the upper bound for m in (12). Complete SHADE algorithm is depicted in pseudo-code below.

Algorithm pseudo-code 2: SHADE

```
1.  Set NP, H and stopping criterion;
2.  G = 0, x_best = {}, k = 1, p_min = 2/NP, A = ∅;
3.  Randomly initialize (1) population P = (x_1,G, …, x_NP,G);
4.  Set M_F and M_CR according to (5);
5.  P_new = {}, x_best = best from population P;
6.  while stopping criterion not met
7.    S_F = ∅, S_CR = ∅;
8.    for i = 1 to NP do
9.      x_i,G = P[i];
10.     r = U[1, H], p_i = U[p_min, 0.2];
11.     Set F_i by (7) and CR_i by (8);
12.     v_i,G by mutation (6);
13.     u_i,G by crossover (3);
14.     if f(u_i,G) < f(x_i,G) then
15.       x_i,G+1 = u_i,G;
16.       x_i,G → A;
17.       F_i → S_F, CR_i → S_CR;
18.     else
19.       x_i,G+1 = x_i,G;
20.     end
21.     if |A|>NP then randomly delete an ind. from A;
22.     x_i,G+1 → P_new;
```

```
    end
    if S_F ≠ ∅ and S_CR ≠ ∅ then
      Update M_F,k (9) and M_CR,k (10), k++;
      if k > H then k = 1, end;
    end
    P = P_new, P_new = {}, x_best = best from population P;
  end
  return x_best as the best found solution
```

4 Network Design and Scoring Methods for Individuals

Since this research paper presents a hybridization of evolutionary computational technique with a complex network, this section describes the network design during an optimization run of the SHADE algorithm and also presents two scoring methods for individuals that were used in the experiment.

Individuals from the SHADE algorithm are represented by nodes in the network and edges mirror the communication between them. Therefore, a new edge is created each time an individual helped to produce a solution of higher quality. The key aspects in the production of a new solution are mutation and crossover steps. Thus all individuals that are present in these steps are candidates for new edges. Once a trial individual $\boldsymbol{u}_i$ succeeds in the selection step, three new edges are created in the network. Those edges connect a trial individual node (its node is also a node for the original individual $\boldsymbol{x}_i$) with its mutation partner's nodes – $\boldsymbol{x}_{pbest}$, $\boldsymbol{x}_{r1}$ and $\boldsymbol{x}_{r2}$. An example of the representation of individuals in a network and example of the creation of new edges after successful mutation is depicted in Figure 1.

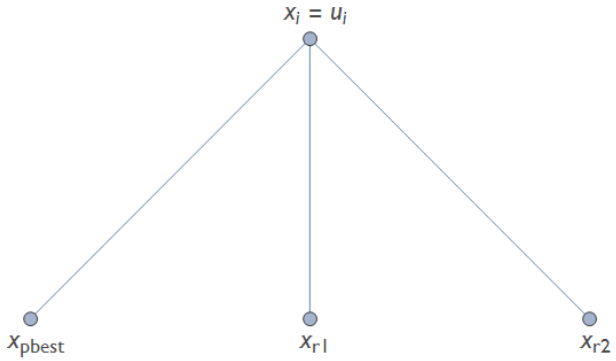

Figure 1: Network edges created for one successful mutation.

A new network is created in each generation to capture the communication dynamic between individuals. Since there can be more than just one exchange of knowledge between the same two individuals in one generation, the network is represented by an undirected multigraph, where the maximum number of edges that can be created during one generation is $3\times NP$, which would happen only if all trial individuals outperformed their original versions.

In order to evaluate the correlation between the quality of an individual and its communication in the complex network, two scoring methods were implemented. First of them depicts the quality of an individual (objective function value score) and is abbreviated to *ofvScore*. This score corresponds to the number of individuals in a population that have worse objective function value than the evaluated individual. Therefore, in an example of the population of 100 individuals, the one with the best objective function value will have *ofvScore* = 99, and the one with the worst objective function value will have *ofvScore* = 0. Second scoring method scores the individual based on their node degree centrality, which is a network feature and the score is abbreviated to *cenScore*. Node degree centrality c_i is equal to the sum of edges that are connected to the *i*-th node. Therefore, the individuals in the population can be scored by their node degree centralities in a similar way as in a case of *ofvScore*. In the same example of a population of 100 individuals, the one with the most connections (edges) in the complex network will have *cenScore* = 99, and the one with least connections will have *cenScore* = 0.

5 Experimental Setting

In this study, a primary assumption, that the individuals of higher quality (better objective function value) are the ones that are driving the evolution towards global optima was tested on a CEC2015 benchmark set of 15 test functions in three distinctive dimensionality settings – 10*D*, 30*D*, and 50*D*. The stopping criterion was set according to the benchmark requirements to $10{,}000\times D$ as well as the number of independent runs – 51. The algorithm settings were – population size $NP = 100$ and historical memory size $H = 10$.

Both scoring methods from the previous section (*ofvScore*, *cenScore*) were recorded as well as the convergence history. To test the assumption, Spearman's rank correlation coefficient was computed between *ofvScore* and *cenScore*, which

should be positive in the case that the assumption is correct – individuals with high *ofvScore* should correspondingly have high *cenScore*.

6 Results and Discussion

In this part, 15 figures with convergence and Spearman's correlation history are depicted for 15 test functions in three-dimensional settings – 10*D*, 30*D*, and 50*D*. Also, the correlation history was classified into three behavior groups – high correlation (H), low correlation (L) and fluctuating correlation (F).

The CEC2015 benchmark set contains four types of test functions – unimodal functions (*f1* and *f2*), simple multimodal functions (*f3* to *f9*), hybrid functions (*f10* to *f12*) and composition functions (*f13* to *f15*). Since the SHADE algorithm uses "current-to-*p*best/1" mutation strategy, the primary assumption of the high correlation between quality of the solution and its knowledge transfer capabilities is incorporated into one of the most important aspects of the algorithm – mutation. The *p*best set of individuals consists of $NP \times p$ best individuals in the population according to their objective function value, and therefore this greedy approach favors them for the production of new mutated vectors. Such behavior is rewarded during the optimization of functions, where the fast convergence is needed – unimodal functions. These serve as an example of the typical high correlation of *ofvScore* and *cenScore* (Figure 1 and Figure 2). It can be seen, that during convergence, the mean correlation stays high somewhere around 0.5 in all dimensionality settings. This can also be found in the case of test functions *f9*, *f12* and *f15*, where SHADE fails to avoid the premature convergence to local optima.

On the other hand, SHADE algorithm has mechanisms, which were designed to prevent the premature convergence on more complex problems. This can be observed on some of the multimodal (*f3*, *f4*, and *f5*) and one composition (*f13*) function. Such behavior is distinctive by a high peak in correlation right at the start of the evolution but quickly decreasing to almost 0 during the later phases while the convergence continues.

A clear fluctuating correlation can be observed in the composition test function *f14*, where there is more than just one peak in the correlation history. In this case, there is a fast convergence to the local optima, which is followed by the slow refining of the final solution in this area, which brings later peaks of correlation.

The remaining multimodal (*f6*, *f7*, and *f8*) and hybrid (*f10* and *f11*) functions experience different correlation behavior in distinctive dimensionality settings.

- *f6* and *f8* – fluctuating behavior in *10D* and 30*D* with the capability of reaching the global optima. High correlation behavior in 50*D*, where the algorithm converges slowly to local optima.
- *f7* – low correlation behavior in *10D* and *30D* with the capability of reaching the global optima. High correlation behavior 50*D*, where the algorithm converges slowly to local optima.
- *F10* – fluctuating behavior in 10*D* and high correlation behavior in 30*D* and 50*D*, with convergence to local optima.
- *F11* - fluctuating behavior in 10*D* with the capability of reaching the global optima. High correlation behavior in 30*D* and 50*D*, with convergence to local optima.

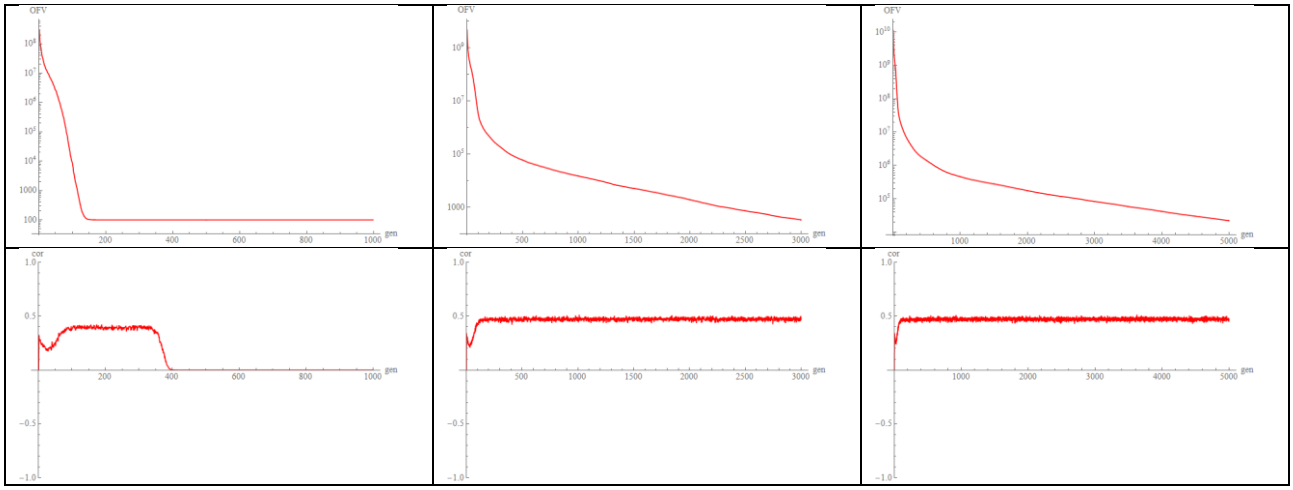

Figure 2: Average convergence graphs and corresponding correlation graphs in 10*D*, 30*D* and 50*D* (from left) for test function *f1*.

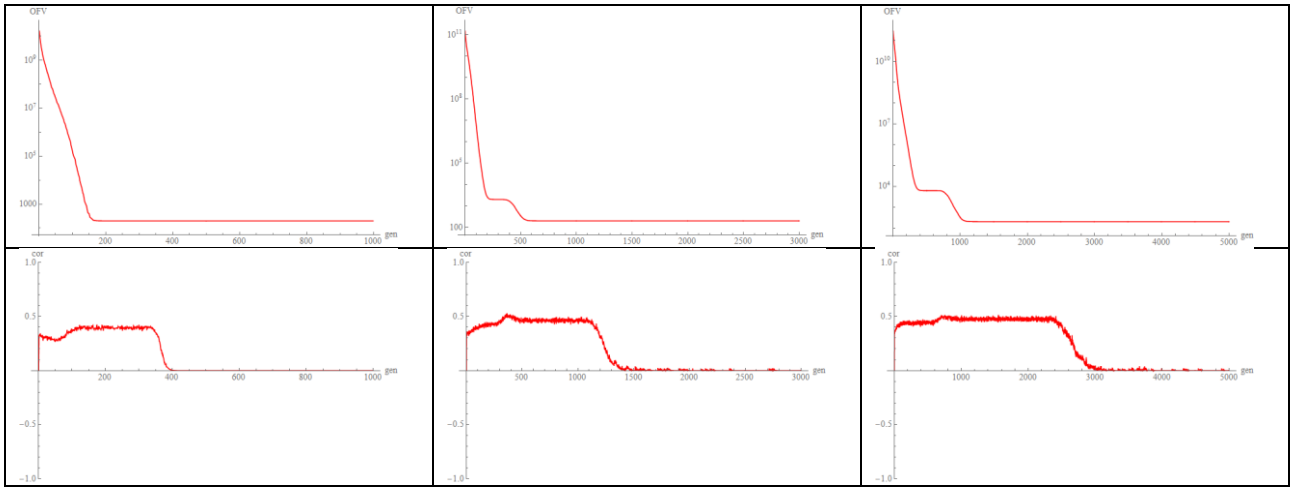

Figure 3: Average convergence graphs and corresponding correlation graphs in 10*D*, 30*D* and 50*D* (from left) for test function *f2*.

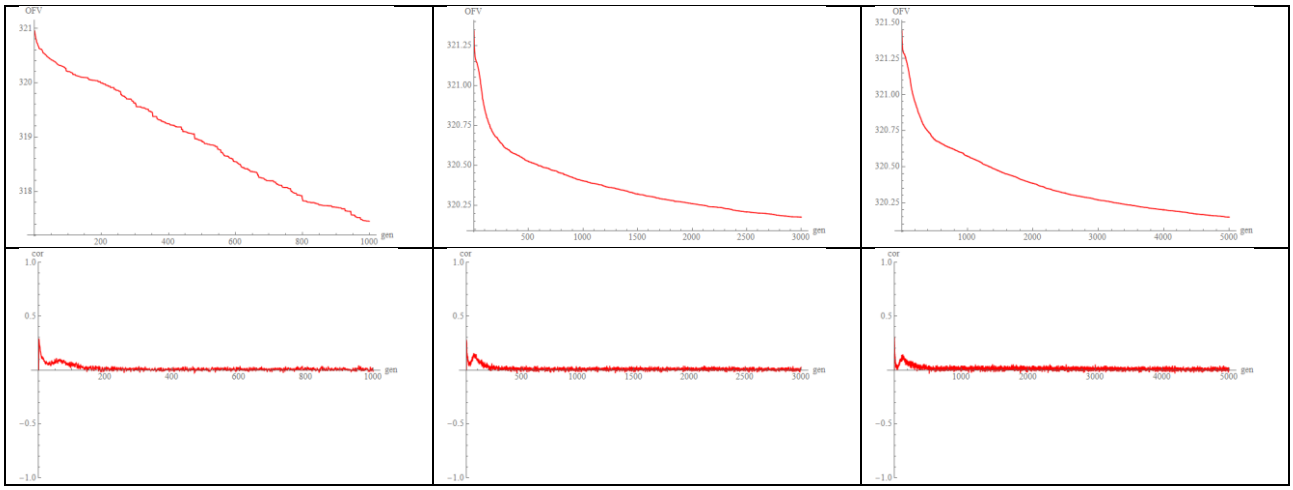

Figure 4: Average convergence graphs and corresponding correlation graphs in 10*D*, 30*D* and 50*D* (from left) for test function *f3*.

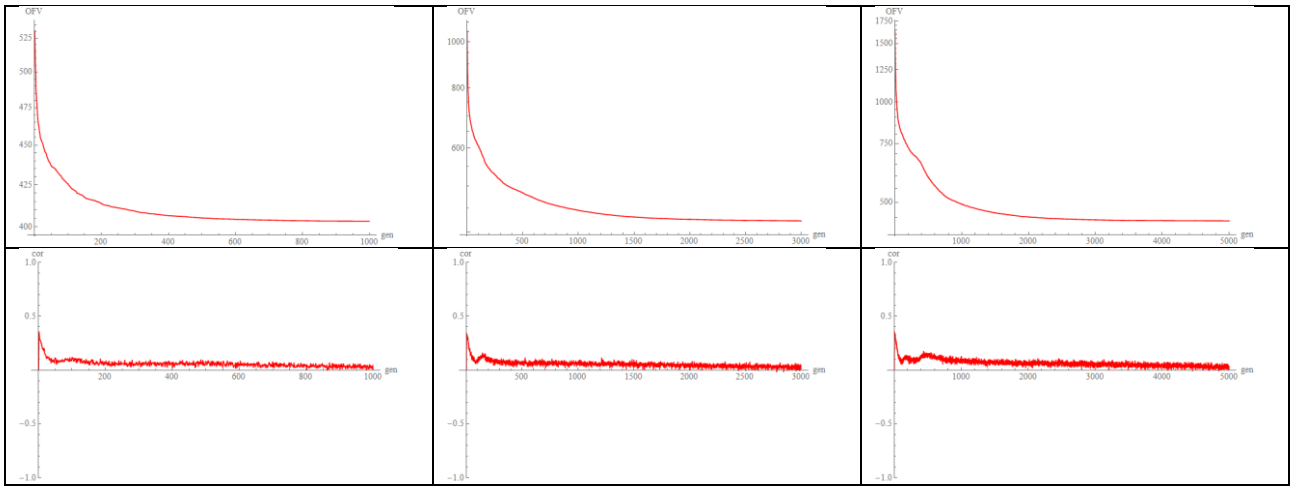

Figure 5: Average convergence graphs and corresponding correlation graphs in 10*D*, 30*D* and 50*D* (from left) for test function *f4*.

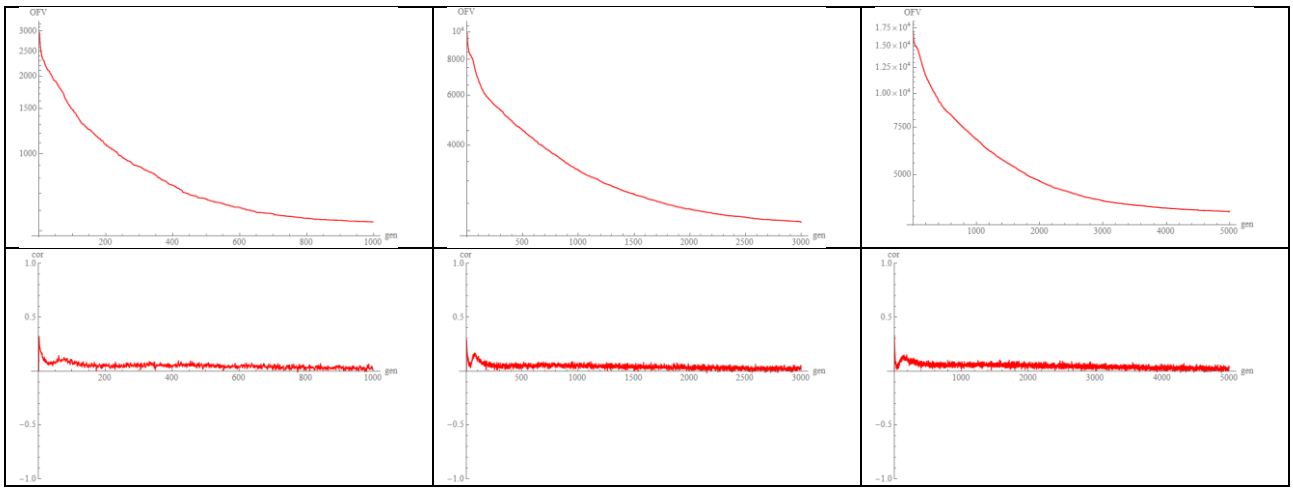

Figure 6: Average convergence graphs and corresponding correlation graphs in 10*D*, 30*D* and 50*D* (from left) for test function *f*5.

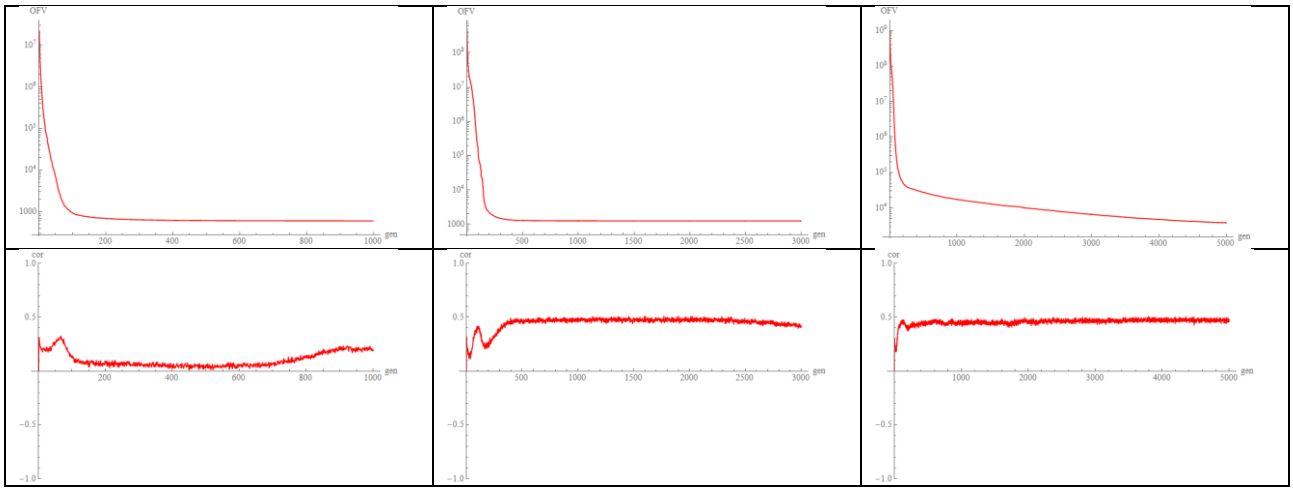

Figure 7: Average convergence graphs and corresponding correlation graphs in 10*D*, 30*D* and 50*D* (from left) for test function *f*6.

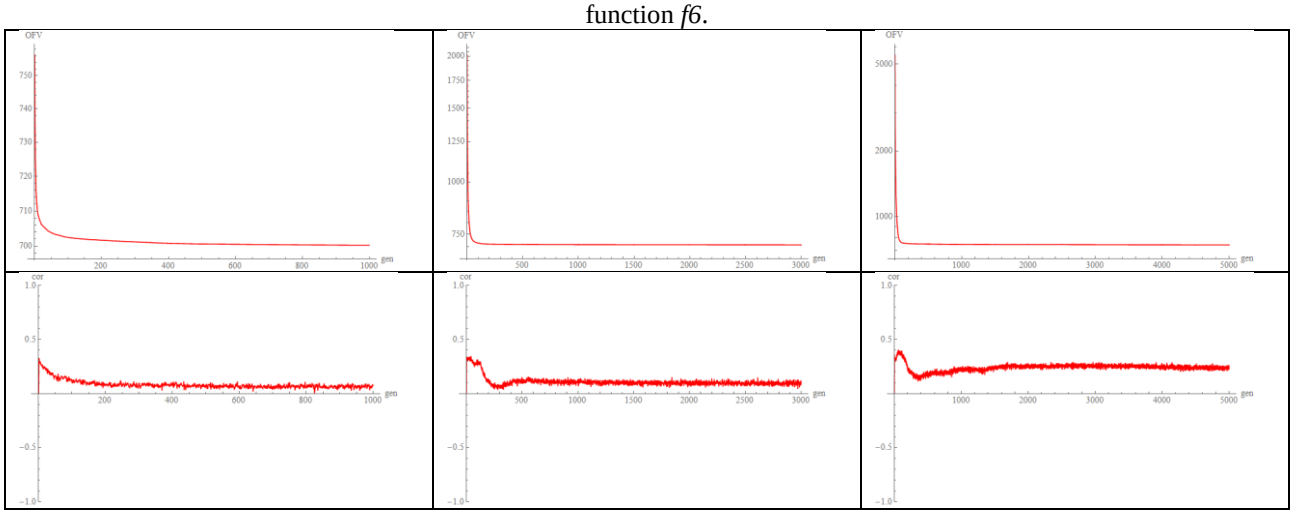

Figure 8: Average convergence graphs and corresponding correlation graphs in 10*D*, 30*D* and 50*D* (from left) for test function *f*7.

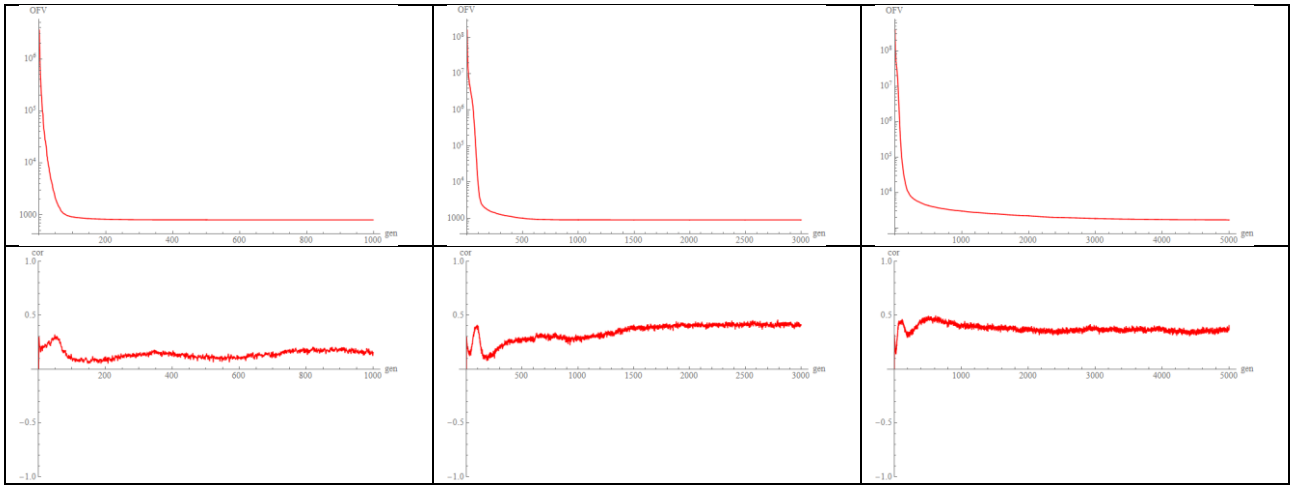

Figure 9: Average convergence graphs and corresponding correlation graphs in 10*D*, 30*D* and 50*D* (from left) for test function *f8*.

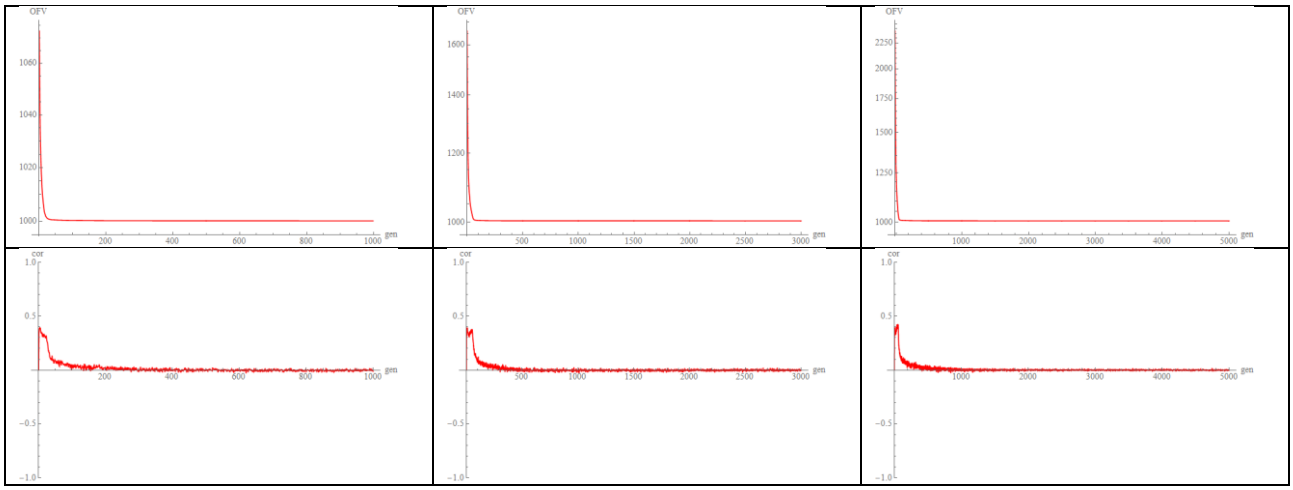

Figure 10: Average convergence graphs and corresponding correlation graphs in 10*D*, 30*D* and 50*D* (from left) for test function *f9*.

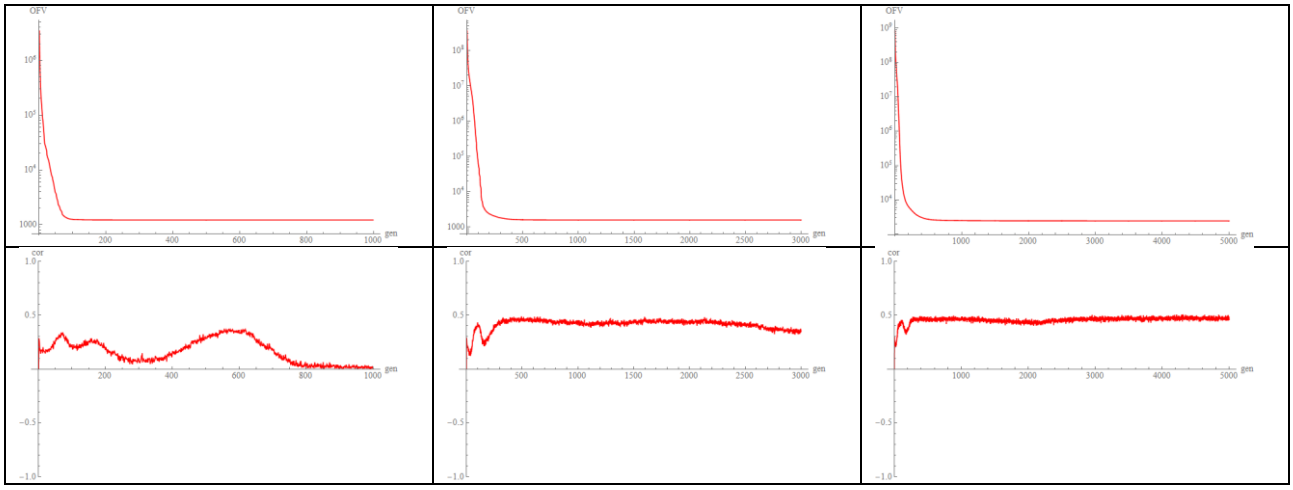

Figure 11: Average convergence graphs and corresponding correlation graphs in 10*D*, 30*D* and 50*D* (from left) for test function *f10*.

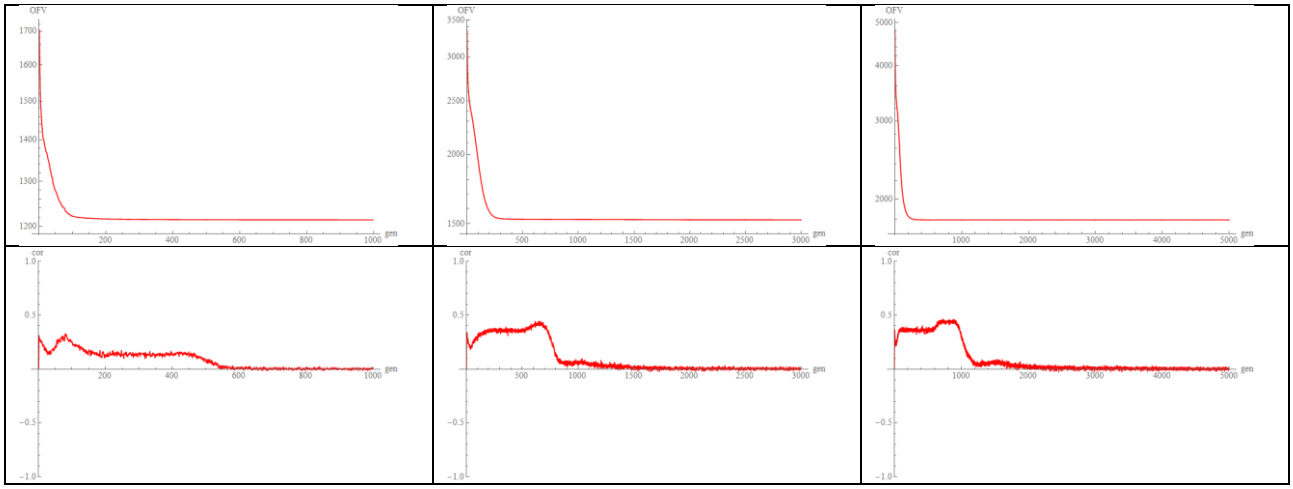

Figure 12: Average convergence graphs and corresponding correlation graphs in 10*D*, 30*D* and 50*D* (from left) for test function *f11*.

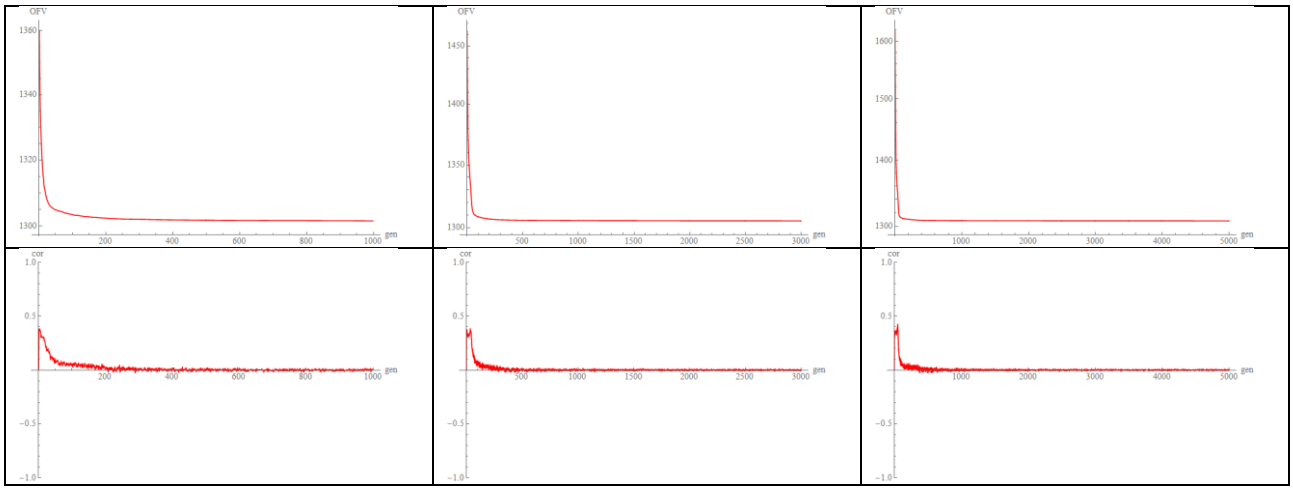

Figure 13: Average convergence graphs and corresponding correlation graphs in 10*D*, 30*D* and 50*D* (from left) for test function *f12*.

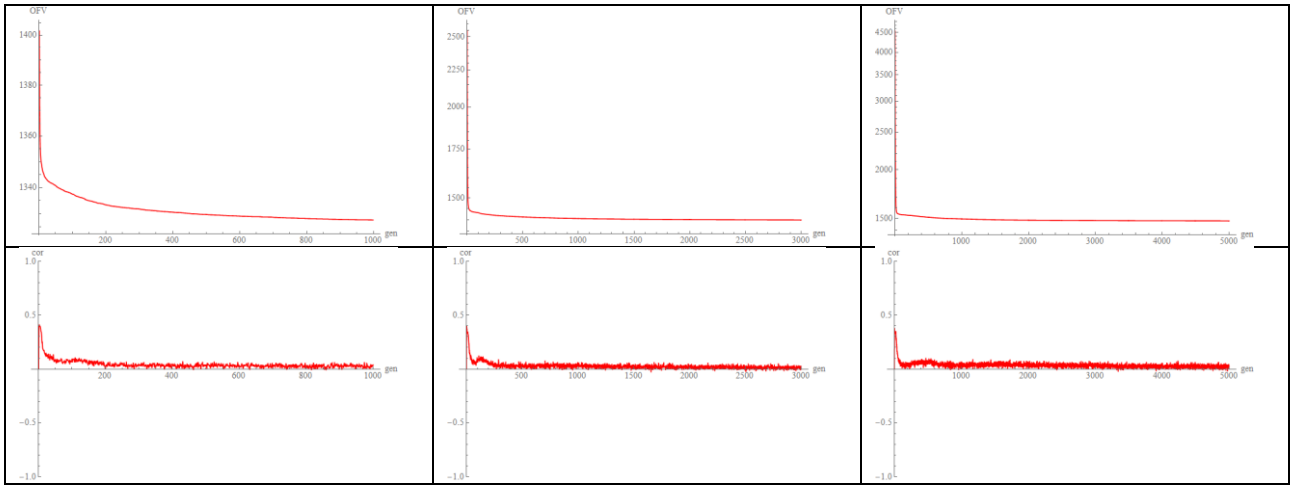

Figure 14: Average convergence graphs and corresponding correlation graphs in 10*D*, 30*D* and 50*D* (from left) for test function *f13*.

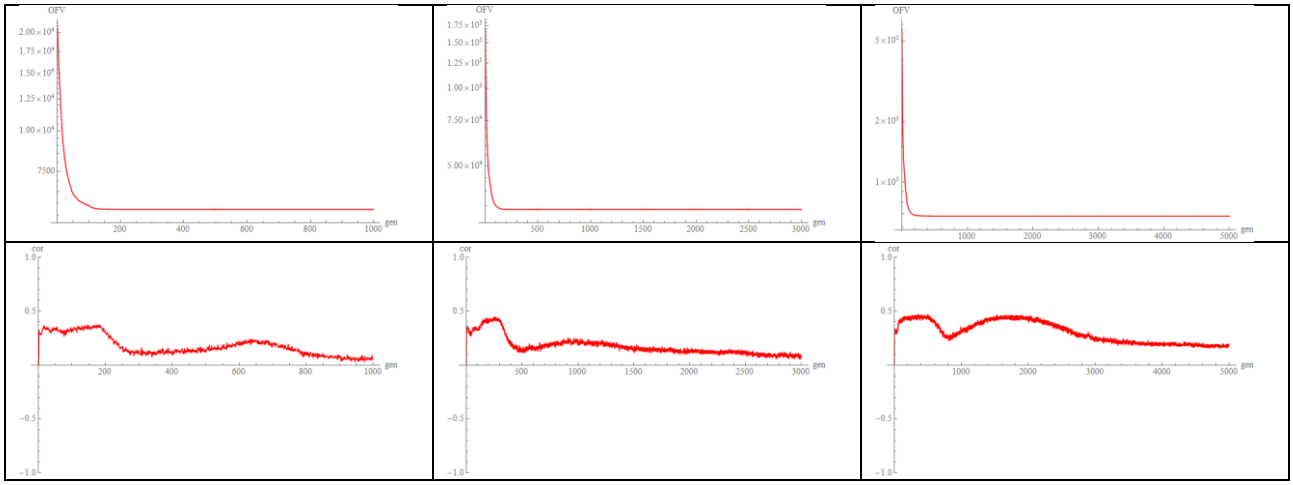

Figure 15: Average convergence graphs and corresponding correlation graphs in 10*D*, 30*D* and 50*D* (from left) for test function *f14*.

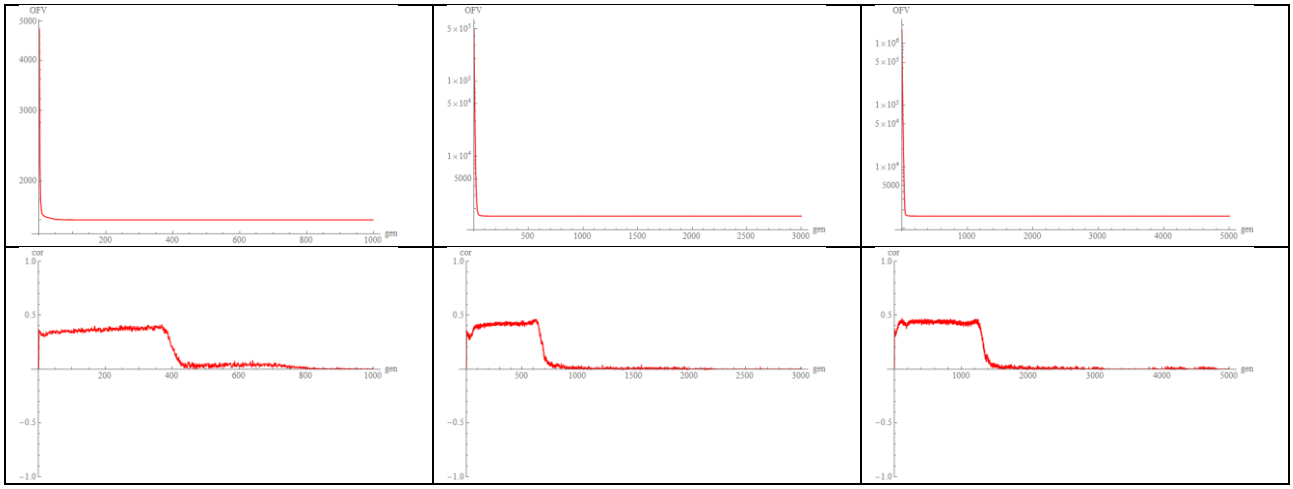

Figure 16: Average convergence graphs and corresponding correlation graphs in 10*D*, 30*D* and 50*D* (from left) for test function *f15*.

The correlation behavior classification is summarized in Table 1, where the cases when SHADE algorithm is not capable of advancing towards the globally optimal solution are highlighted by bold text. It is clear that the ability to obtain globally optimal solution decreases with the dimensionality of the problem and it is invariant to the type of the correlation behavior.

Dim \ *f*	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
10	H	H	**L**	L	L	F	L	F	**L**	**F**	F	**L**	**L**	**F**	**H**
30	H	H	**L**	**L**	**L**	F	L	F	**L**	**H**	**H**	**L**	**L**	**F**	**H**
50	**H**	H	**L**	**L**	**L**	**H**	**H**	**H**	**L**	**H**	**H**	**L**	**L**	**F**	**H**

Table 1: Classification of the correlation behavior in three-dimensional settings (10*D*, 30*D*, and 50*D*), H – high correlation, L – low correlation, F – fluctuating correlation.

The results suggest that the greedy approach used in "current-to-*p*best/1" mutation strategy might not be the most efficient and rather than using a set of best (in terms of objective function value) individuals for the $\boldsymbol{x}_{pbest}$, it could be preferred to use a set given by another common characteristic (potentially one of the network features, e.g. node degree centrality or clustering coefficient). The same goes for a greedy approach to the linear decrease in population size, where the worst individuals are removed from the population during evolution. Findings in this paper suggest that there might be individuals with worse objective function value who might still provide a good search direction. Also, this research supports the idea that there is still much room-for-improvement and the hybrid approach with complex networks is one of the promising candidates for future study.

7 Conclusion

This research presented an analysis of the state-of-the-art DE variant – SHADE with the aid of the complex network and its features, namely – node degree centrality. The analysis was performed on the basis of the CEC2015 benchmark set in

three distinctive dimensional settings (10*D*, 30*D*, and 50*D*) to provide a robust overview of the correlation between the quality of individual solutions and their knowledge transfer capabilities. The findings in this paper suggest that the current greedy approach might not always be the most suitable one and that there is a room-for-improvement. Therefore, the hybridization of the two computational intelligence fields (evolutionary computational techniques and complex networks) is a promising direction for the future research in improving the performance of existing DE-based algorithms, or possibly for the proposals of new hybrid evolutionary algorithms that would incorporate information from complex networks into the algorithm's inner workings.
The future research will aim at the next step in this direction, which is incorporating the information from the complex network into the algorithm to improve its ability to avoid the premature convergence into local optima and to fully use the knowledge transfer capabilities of individual solutions.

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. (1995). Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces (Vol. 3). Berkeley: ICSI.
[2] Storn, R., & Price, K. (1996, May). Minimizing the real functions of the ICEC'96 contest by differential evolution. In Evolutionary Computation, 1996., Proceedings of IEEE International Conference on (pp. 842-844). IEEE.
[3] Price, K. (1997). Differential evolution vs the functions of the 2nd ICEO Evolutionary Computation. In I. EEE International Conference.
[4] P. N. Suganthan, http://www3.ntu.edu.sg/home/epnsugan/index_files/cec-benchmarking.htm
[5] Neri, F., & Tirronen, V. (2010). Recent advances in differential evolution: a survey and experimental analysis. Artificial Intelligence Review, 33(1-2), 61-106.
[6] Das, S., & Suganthan, P. N. (2011). Differential evolution: a survey of the state-of-the-art. IEEE transactions on evolutionary computation, 15(1), 4-31.
[7] Das, S., Mullick, S. S., & Suganthan, P. N. (2016). Recent advances in differential evolution–An updated survey. Swarm and Evolutionary Computation, 27, 1-30.
[8] Gämperle, R., Müller, S. D., & Koumoutsakos, P. (2002). A parameter study for differential evolution. Advances in intelligent systems, fuzzy systems, evolutionary computation, 10, 293-298.
[9] Liu, J., & Lampinen, J. (2002). 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).
[10] Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. IEEE transactions on evolutionary computation, 1(1), 67-82.
[11] Omran, M. G., Salman, A., & Engelbrecht, A. P. (2005). Self-adaptive differential evolution. In Computational intelligence and security (pp. 192-199). Springer Berlin Heidelberg.
[12] Brest, J., Greiner, S., Bošković, B., Mernik, M., & Zumer, V. (2006). Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. Evolutionary Computation, IEEE Transactions on, 10(6), 646-657.
[13] Islam, S. M., Das, S., Ghosh, S., Roy, S., & Suganthan, P. N. (2012). An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 42(2), 482-500.
[14] Qin, A. K., Huang, V. L., & Suganthan, P. N. (2009). Differential evolution algorithm with strategy adaptation for global numerical optimization. Evolutionary Computation, IEEE Transactions on, 13(2), 398-417.
[15] Zhang, J., & Sanderson, A. C. (2009). JADE: adaptive differential evolution with optional external archive. Evolutionary Computation, IEEE Transactions on, 13(5), 945-958.
[16] Tanabe, R., & Fukunaga, A. (2013, June). Success-history based parameter adaptation for differential evolution. In Evolutionary Computation (CEC), 2013 IEEE Congress on (pp. 71-78). IEEE.
[17] Tanabe, R., & Fukunaga, A. S. (2014, July). Improving the search performance of SHADE using linear population size reduction. In Evolutionary Computation (CEC), 2014 IEEE Congress on (pp. 1658-1665). IEEE.
[18] Brest, J., Greiner, S., Boskovic, B., Mernik, M., & Zumer, V. (2006). Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE transactions on evolutionary computation*, *10*(6), 646-657.
[19] Qin, A. K., Huang, V. L., & Suganthan, P. N. (2009). Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE transactions on Evolutionary Computation*, *13*(2), 398-417.
[20] Das, S., Abraham, A., Chakraborty, U. K., & Konar, A. (2009). Differential evolution using a neighborhood-based mutation operator. *IEEE Transactions on Evolutionary Computation*, *13*(3), 526-553.
[21] Mininno, E., Neri, F., Cupertino, F., & Naso, D. (2011). Compact differential evolution. *IEEE Transactions on Evolutionary Computation*, *15*(1), 32-54.

[22] Mallipeddi, R., Suganthan, P. N., Pan, Q. K., & Tasgetiren, M. F. (2011). Differential evolution algorithm with ensemble of parameters and mutation strategies. *Applied Soft Computing*, *11*(2), 1679-1696.
[23] Brest, J., Korošec, P., Šilc, J., Zamuda, A., Bošković, B., & Maučec, M. S. (2013). Differential evolution and differential ant-stigmergy on dynamic optimisation problems. *International Journal of Systems Science*, *44*(4), 663-679.
[24] Brest, J., Maučec, M. S., & Bošković, B. (2016, July). iL-SHADE: Improved L-SHADE algorithm for single objective real-parameter optimization. In *Evolutionary Computation (CEC), 2016 IEEE Congress on* (pp. 1188-1195). IEEE.
[25] Viktorin, A., Pluhacek, M., & Senkerik, R. (2016, July). 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.
[26] Poláková, R., Tvrdík, J., & Bujok, P. (2016, July). L-SHADE with competing strategies applied to CEC2015 learning-based test suite. In *Evolutionary Computation (CEC), 2016 IEEE Congress on* (pp. 4790-4796). IEEE.
[27] Guo, S. M., Tsai, J. S. H., Yang, C. C., & Hsu, P. H. (2015, May). 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.
[28] Awad, N. H., Ali, M. Z., Suganthan, P. N., & Reynolds, R. G. (2016, July). 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.
[29] Brest, J., Maučec, M. S., & Bošković, B. (2017, June). Single objective real-parameter optimization: Algorithm jSO. In Evolutionary Computation (CEC), 2017 IEEE Congress on (pp. 1311-1318). IEEE.
[30] Viktorin, A., Hrabec, D., & Pluhacek, M. (2016). Multi-chaotic differential evolution for vehicle routing problem with profits. In *Proceedings-30th European Conference on Modelling and Simulation, ECMS 2016*. European Council for Modelling and Simulation (ECMS).
[31] Szenkovits, A., Gaskó, N., & Jakab, H. (2016, September). Optimizing Test Input Generation for Reactive Systems with an Adaptive Differential Evolution. In *Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2016 18th International Symposium on* (pp. 214-218). IEEE.
[32] Zamuda, A., Sosa, J. D. H., & Adler, L. (2016). Constrained differential evolution optimization for underwater glider path planning in sub-mesoscale eddy sampling. *Applied Soft Computing*, *42*, 93-118.
[33] Ekici, B., Chatzikonstantinou, I., Sariyildiz, S., Tasgetiren, M. F., & Pan, Q. K. (2016, July). A multi-objective self-adaptive differential evolution algorithm for conceptual high-rise building design. In *Evolutionary Computation (CEC), 2016 IEEE Congress on* (pp. 2272-2279). IEEE.
[34] D. A. Dombo and K. A. Folly, "Self-adaptive differential evolution based power system stabilizers," 2017 IEEE Symposium Series on Computational Intelligence (SSCI), Honolulu, HI, USA, 2017, pp. 1-6. doi: 10.1109/SSCI.2017.8285412
[35] Leon, M., Zenlander, Y., Xiong, N., & Herrera, F.. Designing optimal harmonic filters in power systems using greedy adaptive Differential Evolution. In Emerging Technologies and Factory Automation (ETFA), 2016 IEEE 21st International Conference on (pp. 1-7). IEEE.
[36] Karafotias, G., Hoogendoorn, M., & Eiben, Á. E. (2015). Parameter control in evolutionary algorithms: Trends and challenges. *IEEE Transactions on Evolutionary Computation*, *19*(2), 167-187.
[37] Zamuda, A., & Brest, J. (2015). Self-adaptive control parameters′ randomization frequency and propagations in differential evolution. *Swarm and Evolutionary Computation*, *25*, 72-99.
[38] Tanabe, R., & Fukunaga, A. (2016, September). How Far Are We from an Optimal, Adaptive DE?. In *International Conference on Parallel Problem Solving from Nature* (pp. 145-155). Springer International Publishing.
[39] Skanderova, L., Fabian, T., Zelinka, I. (2016). Small-world hidden in differential evolution. In *Evolutionary Computation (CEC), 2016 IEEE Congress on* (pp. 3354-3361).
[40] Zelinka I, Davendra D, Lampinen J, Senkerik R, Pluhacek M Evolutionary algorithms dynamics and its hidden complex network structures. In: *Evolutionary Computation (CEC), 2014 IEEE Congress on*, 2014, pp 3246-3251.
[41] Gajdos P, Kromer P, Zelinka I Network Visualization of Population Dynamics in the Differential Evolution. In: *Computational Intelligence, 2015 IEEE Symposium Series on*, 2015. pp 1522-1528.
[42] Skanderova L, Fabian T (2015) Differential evolution dynamics analysis by complex networks. *Soft Computing*:1-15.
[43] Metlicka M, Davendra D Ensemble centralities based adaptive Artificial Bee algorithm. In: *Evolutionary Computation (CEC), 2015 IEEE Congress on*, 2015. pp 3370-3376.
[44] Tomaszek, L., & Zelinka, I. (2016). On performance improvement of the SOMA swarm based algorithm and its complex network duality. In *Evolutionary Computation (CEC), 2016 IEEE Congress on* (pp. 4494-4500).
[45] Viktorin, A., Senkerik, R., Pluhacek, M., & Kadavy, T. (2017). Towards better population sizing for differential evolution through active population analysis with complex network. In *Conference on Complex, Intelligent, and Software Intensive Systems* (pp. 225-235).
[46] Viktorin, A., Pluhacek, M., & Senkerik, R. (2016, September). Network Based Linear Population Size Reduction in SHADE. In *Intelligent Networking and Collaborative Systems (INCoS), 2016 International Conference on* (pp. 86-93). IEEE.