Differential evolution based on node strength

Lenka Skanderova* and Tomas Fabian

Department of Computer Science,  
Faculty of Electrical Engineering and Computer Science,  
VSB – Technical University of Ostrava,  
Ostrava, Czech Republic  
Fax: +420-59-6919597  
Email: lenka.skanderova@vsb.cz  
Email: tomas.fabian@vsb.cz  
*Corresponding author

Ivan Zelinka

Modeling Evolutionary Algorithms Simulation and Artificial Intelligence,  
Faculty of Electrical and Electronics Engineering,  
Ton Duc Thang University,  
Ho Chi Minh, Vietnam  
Email: ivan.zelinka@tdt.vn.edu  
and  
Department of Computer Science,  
Faculty of Electrical Engineering and Computer Science,  
VSB – Technical University of Ostrava,  
Ostrava, Czech Republic  
Fax: +420-59-6919597  
Email: ivan.zelinka@vsb.cz

Abstract: In this paper, three novel algorithms for optimisation based on the differential evolution algorithm are devised. The main idea behind those algorithms stems from the observation that differential evolution dynamics can be modelled via complex networks. In our approach, the individuals of the population are modelled by the nodes and the relationships between them by the directed lines of the graph. Subsequent analysis of non-trivial topological features further influence the process of parent selection in the mutation step and replace the traditional approach which is not reflecting the complex relationships between individuals in the population during evolution. This approach represents a general framework which can be applied to various kinds of differential evolution algorithms. We have incorporated this framework with the three well-performing variants of differential evolution algorithms to demonstrate the effectiveness of our contribution with respect to the convergence rate. Two well-known benchmark sets (including 49 functions) are used to evaluate the performance of the proposed algorithms. Experimental results and statistical analysis indicate that the enhanced algorithms perform better or at least comparable to their original versions.

Keywords: differential evolution dynamics; complex network; node strength; hybrid mutation operator; self-adapting parameter.


Biographical notes: Lenka Skanderova is a PhD student at the Technical University of Ostrava in the Czech Republic. Her research interests are related to artificial intelligence, especially evolutionary and swarm algorithms, chaos, and complex networks. She has published research papers at national and international journals, conference proceedings and book chapters.

Tomas Fabian received his PhD from the Technical University of Ostrava and currently he works as an Assistant Professor at the Department of Computer Science, Technical University of Ostrava. His research is focused on the computer graphics and vision-based algorithms.
1 Introduction

Differential evolution (DE) is an efficient population-based evolutionary algorithm for optimisation introduced by Storn and Price (1995). Despite its good convergence properties and the fact that it is easy to implement, DE has some disadvantages: in some cases, its convergence is unstable and it easily drops into the local extreme (Wu et al., 2011). To eliminate these bottlenecks is the main goal of many scientific publications.

Chiou and Wang (1998) introduced a hybrid DE, where two additional operations, i.e., accelerated phase and migrating phase, have been embedded into the original version of the DE to improve the convergence rate without reducing the diversity among the individuals.

Lampinen et al. (2001) described the first version of a generalised differential evolution (GDE) extending the original version of DE for constrained multi-objective optimisation by modifying the selection rule of the original DE. Three years later, Kukkonen and Lampinen (2004) introduced GDE2, which is a modification of GDE based again on the selection operation. The main bottleneck of both versions is that these algorithms are very sensitive to the settings of the control parameter values. The third version of GDE presented by Kukkonen and Lampinen (2005) modifies the earlier GDE versions by using a growing population and non-dominated sorting with pruning of non-dominated solutions to decrease the population size at the end of each generation. This modification improves the diversity and makes the method more stable for the selection of control parameter values, moreover, the requirements to the number of function evaluations has been significantly reduced.

Brest et al. (2006) described DE using self-adaptive control parameters (jDE), where each individual in the population is extended with its own control parameters \( F \) (scale factor) and \( C_R \) (crossover rate). To improve the properties of DE, Yang et al. (2008) incorporated the neighbourhood search (NS) into DE algorithm and Zhenyu et al. (2008) proposed the self-adaptive DE with NS.

In 2011, Wang et al. introduced the DE with composite trial vector generation strategies and control parameters, where authors use the strategy candidate pool with three variants of DE – DE/rand/1/bin, DE/rand/2/bin and DE/current-to-rand. For each target vector three trial vectors are generated and the best one is then selected to the next generation, if it is better than a target vector.

Zhou et al. (2013) introduced the DE with intersect mutation operator denoted as IMDE. In this algorithm, the population of individuals is divided into two parts – better and worse according to their objective function values. Authors described two novel mutation and crossover operations to generate new individuals.

In 2015, Yi et al. presented a new DE with hybrid mutation operator and self-adapting control parameters for global optimisation (HSDE), where each individual is enhanced by its own scale factor \( F \) and crossover rate \( C_R \) as in the jDE. Then for each target vector a mutation vector is created such that five mutually different parents are selected randomly. If the objective function value of a target vector is better than the objective function value of the first two selected parents the classical DE/rand/1/bin strategy is applied to generate mutation vector, otherwise DE/current-to-best is applied.

The most of the above mentioned researches are dealing with three ways how to improve DE. The first one is to propose more efficient mutation operators, the second one is focused on the control parameters settings and the third method deals with the process of the parents selection in the mutation step. In this work, we are focused on the last mentioned method – parents selection in the mutation step. We use modelling of the DE dynamics by complex networks. The main motivation is to better understand relationships between individuals and using of this knowledge to improve the DE convergence rate.

The rest of the paper is organised as follows: The DE and the related work are described briefly in Sections 2 and 3, respectively. The goal of Section 4 is to clarify the creation of the complex network by the DE dynamics and to describe the role of the node strength in the presented principle of the parent selection in the mutation step. Experiments and their results are provided in Section 5 and they are discussed in Section 6.

2 Background

DE works with a population of NP solution vectors (individuals) \( x_i^G \), where \( i = \{1, \ldots, NP\} \) and \( G \) denotes a
Each solution vector \( \vec{x} \) consists of \( D \) parameters. Each parameter is constrained by its search range \([x_{\text{lower}}, x_{\text{upper}}]\), where lower and upper denotes lower and upper bound for each parameter and \( j = \{1, \ldots, D\} \) is an index of a parameter.

The first population is generated randomly in the space of possible solutions. Then for each target vector \( \vec{x} \), three mutually different solution vectors (parents) \( \vec{x}_1, \vec{x}_2, \vec{x}_3 \), and \( \vec{x}_n \) are selected randomly to generate a mutation vector \( \vec{v} \) as follows:

\[
v = \vec{x}_i - \vec{x}_j + F ( \vec{x}_m - \vec{x}_n ),
\]

where \( F \) is the scale parameter. It is commonly known that the scale factor \( F \) is strictly greater than zero and as mentioned in Mallipeddi et al. (2011), it is usually chosen in interval \([0.5, 1.0]\). The values of \( F \) smaller than 0.4 and greater than 1.0 are occasionally effective (Storn and Price, 1995). On the other hand, a larger \( F \) increases the probability of escaping from a local optimum (Gámperle et al., 2002). According to Ronkkonen et al. (2005), typical values of \( F \) are between 0.4 and 0.95.

In the crossover operation, parameters of a mutation vector are combined with parameters of a target vector on the basis of the crossover rate \( CR \) to generate a trial vector \( \vec{u} \). This operation can be mathematically defined as follows:

\[
\vec{u}_{ij} = \begin{cases} 
\vec{v}_{ij} & \text{if } r(j) \leq CR \text{ or } j = mn(i) \\
\vec{x}_{ij} & \text{if } r(j) > CR \text{ and } j \neq mn(i),
\end{cases}
\]

where \( CR \in [0, 1] \), \( mn(i) \in \{1, \ldots, D\} \) is an integer selected randomly with the uniform distribution ensuring that a trial vector \( \vec{u} \) will contain at least one parameter from a mutation vector \( \vec{v} \), and \( r_j \in [0, 1] \) represents a real number randomly generated with the uniform distribution from the unit interval for each \( j \) parameter (Storn and Price, 1995).

According to Mallipeddi et al. (2011), a large value of \( CR \) speeds up convergence rate. Storn and Price (1995) suggested \( CR = 0.1 \) as a good initial choice. On the other hand, \( CR = 0.9 \) or 1.0 can be used to increase the convergence speed. Gämperle et al. (2002) consider a good choice for \( CR \) to be between 0.3 and 0.9. The number of trial solutions will be reduced when \( CR = 1.0 \). Therefore, \( CR = 0.9 \) or \( CR = 0.99 \) can be used instead. Price et al. (2006) and Ronkkonen et al. (2005) emphasise that for separable problems \( CR \) from the range \([0, 0.2]\) is the best while for multi-modal parameters dependent problems \( CR \) in the range \([0.9, 1.0]\) is the best.

The crossover operation is followed by selection operation. If an objective function value of a trial vector \( f(\vec{u}) \) is better than an objective function value of a target vector \( f(\vec{x}) \), a trial vector (offspring) will survive to the next generation, otherwise a target vector will survive, see the following equation:

\[
\vec{x}_{i+1}^{G} = \begin{cases} 
\vec{u}_{i}^{G} & \text{if } f(\vec{u}_{i}^{G}) \leq f(\vec{x}_{i}^{G}) \text{ (for min. problem)} \\
\vec{x}_{i}^{G} & \text{otherwise.}
\end{cases}
\]

### 3 Related work

The idea to visualise the evolutionary algorithm dynamics by complex networks has been proposed by Zelinka et al. (2010), where DE and self-organising migrating algorithm (SOMA) have been investigated. In this study, each individual is represented by a node and the directed lines between these nodes reflect the dynamics in the population, i.e., interactions between individuals. In the case of the DE, only individuals, that have been replaced by better offspring are recorded like the nodes with added connections.

The authors omitted the philosophy that a worse parent is replaced by a better offspring. The interpretation that an individual (worse parent) is moving to a better position (better offspring) has been accepted. In other words, if a target vector is replaced by a trial vector in the next generation, it is considered to be an activation of a target vector by three solution vectors (selected in a mutation operation to generate a mutation vector), see Figure 1.

**Figure 1** Activation of a target vector \( \vec{x} \) by solution vectors \( \vec{x}_1, \vec{x}_2, \vec{x}_3 \) represented by a directed graph

Zelinka et al. (2012) extended this study by the visualisation of the CN dynamics using coupled map lattices (CML), where the CNs are converted to the CML systems for control and analysis of network dynamics.

Two years later, Davendra et al. (2014) analysed the development of a CN created on the basis of the discrete self-organising migrating algorithm (DSOMA) dynamics, where the flow-shop scheduling with blocking constraint problem has been used as the test problem. As well as in the works of Zelinka et al. (2010, 2011), the individuals are represented by the nodes and the relationships between them by arcs of the CN, and the attributes as adjacency graph, minimal cut, degree centrality, closeness centrality, betweenness centrality, Katz centrality, mean neighbour degree, k-Clique, k-Plan, k-Club, k-Clan, and community graph plots are analysed. On the basis of the results, authors concluded that the DSOMA population does behave like a CN and for this reason it can be analysed as such.

The above mentioned works deal with the CN creation by the evolutionary algorithm dynamics and analysis of the properties of such created CNs, however, we have found
that no research exists regarding the incorporation of the analysed properties of the CN back into the evolutionary algorithms for a purpose to improve the performance of the corresponding evolutionary algorithm.

In the following sections, we will introduce the approach based on the idea to incorporate the strength of the node of the CN representing the individual in the population back into the DE algorithms with the goal to improve the DE convergence rate.

4 DE creates complex networks

In this section, we would like to describe in detail the principle of the complex network creation by the DE dynamics. Regardless we test our approach in three different variants of DE, the principle of the CN creation is the same for all of them. The motivation behind this step is to model the dynamics of DE by the CNs to better understand the relationships between individuals in the population and to incorporate the modified principle of the parent selection in the mutation step in DE that takes into account usefulness of the individual in the population. This usefulness is given by the strength of the node in CN which represents the individual in the population.

Kim and Wilhelm (2008) describe the CN as the graph with non-trivial topological features, i.e., features that do not occur in the simple networks but occur in the real networks. In accordance with Boccaletti et al. (2006), the CN can be represented by the directed or undirected graph $G = (V, E)$ where $V = \{v_1, \ldots, v_N\}$ denotes the set of vertices (nodes) and $E = \{e_1, \ldots, e_K\}$ is the set of the edges.

4.1 Graph creation on the basis of DE dynamics

In this subsection, we will describe the way how the graph is constructed by the dynamics of DE. We will generate one graph for each generation of the DE algorithm. Due to this mechanism, we will be able to track the development of the relationships between members of the population.

At the beginning of the algorithm, the initial population is generated randomly with the uniform distribution in the space of possible solutions as usual and the zero weighted adjacency matrix $W^G (G$ denotes the actual generation) capturing the weighted adjacency of target vectors and corresponding solution vectors selected in the mutation operation (to generate a mutation vector) is generated.

Then for each target vector $\bar{x}_n^G$ three random solution vectors $\bar{x}_n^G, \bar{x}_n^G, \text{and} \bar{x}_n^G$ are chosen and a mutation vector $\bar{v}_n^G$ and then a trial vector $\bar{u}_n^G$ is generated in the accordance with equations (1), and (2), respectively.

If $f(\bar{u}_n^G) \leq f(\bar{x}_n^G)$ then three arcs will be created such that a node representing a randomly selected solution vector $\bar{x}_n^G$ will be a source and a node representing a target vector will be a destination of the arc, see Figure 1. And the weighted adjacency matrix will be recomputed such that

$W_{r_j,i}^G = W_{r_j,i}^G + 1$, where $r_j \in \{1, 2, 3\}$ is the row and $i$ is the column of the weighted adjacency matrix $W^G$. If $f(\bar{u}_n^G) > f(\bar{x}_n^G)$ no arc will be created and the weighted adjacency matrix $W^G$ remains unchanged. For better illustration of the graph creation on the basis of the DE dynamics see Figure 2.

4.2 Degree centrality and node strength

Freeman (1979) introduced the idea that the degree of a node $i$ is the number of nodes that this node is connected to.
This measure can be formally described by the following equation:

\[ k_i = C_D(i) = \sum_{j=1}^{N} x_{ij} \]  
(4)

where \( i \) denotes the focal node, \( j \) represents all other nodes, \( N \) denotes the total number of nodes in the graph and \( x_{ij} \) is the element of the adjacency matrix defined as 1 if the node \( i \) is connected to node \( j \), and 0 otherwise.

This equation has been extended for the analysis of the weighted networks (Barrat et al., 2004; Newman, 2004; Opsahl et al., 2008) to the following equation:

\[ s_i = C_D^w(i) = \sum_{j=1}^{N} W_{ij} \]  
(5)

where \( W_{ij} \) is the element of the weighted adjacency matrix, where \( W_{ij} > 0 \) if the node \( i \) is connected to node \( j \). The value of \( W_{ij} \) represents the weight of the edge. This measure has been labelled node strength.

In directed networks, two types of degree can be distinguished – in-degree \( k_i^\text{in} \) denoting the number of edges that are directed towards a focal node and out-degree \( k_i^\text{out} \) representing the number of edges originating from this node. In the case of the weighted directed network, the node strength is divided into node in-strength \( s_i^\text{in} \) and out-strength \( s_i^\text{out} \). Because we will work with the weighted oriented graphs, we will distinguish in-strength and out-strength of the node and the node strength will be computed according to the following equation:

\[ s_i = \sum_{j=1}^{N} W_{ij} + \sum_{j=1}^{N} W_{ji} \]  
(6)

where \( W \) represents the weighted adjacency matrix and \( N \) the number of nodes in the CN that corresponds with the number of individuals in the population.

4.3 Incorporation of node strength to parent selection in mutation step

The complex networks seem to be an appropriate tool to visualise the relationships between individuals during the evolution of the population in evolutionary algorithms. In this work, individuals are modelled by nodes and relationships between them by arcs of the graph. As mentioned above, only relationships between target vectors activated to move to a better position and their activators will be modelled.

For each node of the corresponding CN the node strength is computed according to equation (6). Each individual of the population has its own parameter ‘strength’, where the strength of the corresponding node is recorded. The higher value of ‘strength’ parameter means the higher probability to be selected as the parent in the mutation step in the DE. The probability of selection is computed according to the following equation:

\[ p_i = \frac{s_i^\text{in}}{\sum_{j=1}^{N} s_j^\text{in}} \]  
(7)

Algorithm 1  DE/deg/1/bin

\{ \[ D \ldots \text{problem dimension}, \ NP \ldots \text{population size} \] \}
\{ \[ F \ldots \text{scale factor}, \ CR \ldots \text{crossover rate} \] \}
\{ \[ W \ldots \text{weighted adjacency matrix}, \] \}
\{ \[ \text{sumSTR} \ldots \text{the sum of all node strengths in } W \} \}
\{-\text{Initialisation}-\}
\{\text{Generate uniformly distributed random population within search space}\}

Create the empty weighted adjacency matrix \( W \)

while stopping criteria is not met do

for \( i = 1 \) to \( NP \) do

if \( \text{sumSTR} > 0 \) then

Randomly select indexes \( r_1, r_2, \) and \( r_3 \) on the basis of the property node strength. The individual with the higher value of the node strength has the higher probability to become the parent.

else

Randomly select indexes \( r_1, r_2, \) and \( r_3 \) \((r_1 \neq r_2 \neq r_3 \neq i)\)

end if

\[ \bar{v}_{ij} = \bar{x}_{ij}^G + F (\bar{x}_{ij}^G - \bar{x}^G_i) \]

\( \text{rand} \{1, \ldots, D\} \)

for \( j = 1 \) to \( D \) do

if \( r(j) \leq CR \) or \( j = \text{rand} \) then

\[ u_{ij} = \bar{v}_{ij} \]

else

\[ u_{ij} = \bar{x}_{ij}^G \]

end if

end for

if \( f(\bar{u}^G) \leq f(\bar{x}^G) \) then

\[ \bar{x}_{ij}^{G+1} = \bar{u}_{ij}^G \]

\[ W_{ji}^G = W_{ji}^G + 1, \ W_{ij}^G = W_{ij}^G + 1, \text{ and} \]

\[ W_{ij}^G = W_{ij}^G + 1 \]

else

\[ x_{ij}^{G+1} = x_{ij}^G \]

end if

end for

For each individual \( \bar{x}_{ij}^{G+1} \) recompute node strength according to equation (6)

For each individual \( \bar{x}_{ij}^{G+1} \) recompute the probability of selection \( p_i \) according to the equation (7)

end while
This means that in the mutation operation, solution vectors $x_{g}^{0}$, $x_{g}^{1}$, $x_{g}^{2}$ are selected on the basis of equation (7) (roulette wheel selection).

At the beginning of the algorithm or in the case that no arc has been created, it is not possible to select the solution vectors in the mutation operation on the basis of the parameter 'strength'. In such case, the individuals are selected as in the original DE algorithm. For this reason, it is necessary to test, if the weighted adjacency matrix is not zero matrix, this can be ensured for example by the sum of all members of the adjacency matrix, which is in Algorithm 1 ensured by the variable $\text{sumSTR}$.

The novel algorithm based on the DE/rand/1/bin will be denoted as DE/deg/1/bin in the following text.

5 Experimental results

We have selected three variants of DE to be enhanced by our approach of the parents selection in the mutation step – DE/rand/1/bin, jDE, and CoDE and tree novel algorithms have been developed – DE/deg/1/bin, jDE/deg and CoDE/deg. The algorithms are implemented in C#, .NET Framework 4.5.1 and run on a computer with CPU Intel(R) Xeon(R) 2.83 GHz. To evaluate the performance of the presented algorithms, 21 benchmark functions from Yao et al. (1999) and 28 benchmark functions from the CEC 2013 (Liang et al., 2013a) have been used.

5.1 Introduction of the benchmark sets

To evaluate the performance of our enhanced variants of DE, two well-known benchmark sets are selected and the enhanced algorithms are compared with their original versions.

5.1.1 Benchmark set 1 and parameters setting

The benchmark set 1 contains 21 testing functions proposed by Yao et al. (1999). The functions can be divided into several categories:

- high-dimensional problems: $f_{1}$–$f_{13}$
- uni-modal functions: $f_{1}$–$f_{5}$
- discontinuous function with one minimum: $f_{6}$
- noisy quartic function: $f_{7}$
- multi-modal functions, where the number of local minima increases exponentially with the problem dimension: $f_{8}$–$f_{13}$
- low-dimensional functions which have only few local minima: $f_{14}$–$f_{21}$.

For all three algorithms, the population size has been set to $NP = 100$. The maximum number of generations for all benchmark functions have been selected according to Zhou et al. (2013) except the function $f_{6}$, where the maximum number of generations has been set to be just 2,000. The initial population is generated randomly with the uniform distribution in the ranges mentioned in Yao et al. (1999).

5.1.2 Benchmark set 2 and parameters setting

Benchmark set 2 consists of 28 benchmark functions from CEC 2013 Special session on real-parameter optimisation. A detailed description of the 28 benchmark functions can be found in Liang et al. (2013a). These benchmark functions can be divided into three classes:

- uni-modal functions: $f_{1}$–$f_{5}$
- multi-modal functions: $f_{6}$–$f_{20}$
- composition functions: $f_{21}$–$f_{28}$.

In the benchmark set 2, the number of problem dimensions is set to 30 for all 28 test functions. The detailed parameter settings for benchmark set 2 is given as follows: The population size is set to 100 for all proposed algorithms in each benchmark function. We have not worked with the maximum number of the objective function evaluation as suggested in Liang et al. (2013b) because we need to create the same number of the complex networks for all selected state-of the-art algorithms and their enhanced versions.

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Experimental results for the benchmark set 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Funct.</td>
<td>Mean</td>
</tr>
<tr>
<td>$f_{1}$</td>
<td>5.91E-14</td>
</tr>
<tr>
<td>$f_{2}$</td>
<td>5.26E-10</td>
</tr>
<tr>
<td>$f_{3}$</td>
<td>1.70E-24</td>
</tr>
<tr>
<td>$f_{4}$</td>
<td>4.32E-02</td>
</tr>
<tr>
<td>$f_{5}$</td>
<td>5.15E-12</td>
</tr>
<tr>
<td>$f_{6}$</td>
<td>30.982</td>
</tr>
<tr>
<td>$f_{7}$</td>
<td>4.44E-03</td>
</tr>
<tr>
<td>$f_{8}$</td>
<td>-5.467,795</td>
</tr>
<tr>
<td>$f_{9}$</td>
<td>8.18E+01</td>
</tr>
<tr>
<td>$f_{10}$</td>
<td>6.27E-08</td>
</tr>
<tr>
<td>$f_{11}$</td>
<td>1.48E-04</td>
</tr>
<tr>
<td>$f_{12}$</td>
<td>1.05E-14</td>
</tr>
<tr>
<td>$f_{13}$</td>
<td>1.39E-13</td>
</tr>
<tr>
<td>$f_{14}$</td>
<td>0.9980038</td>
</tr>
<tr>
<td>$f_{15}$</td>
<td>2.64E-05</td>
</tr>
<tr>
<td>$f_{16}$</td>
<td>-1.0316285</td>
</tr>
<tr>
<td>$f_{17}$</td>
<td>0.398</td>
</tr>
<tr>
<td>$f_{18}$</td>
<td>3.00E+00</td>
</tr>
<tr>
<td>$f_{19}$</td>
<td>-10.1532</td>
</tr>
<tr>
<td>$f_{20}$</td>
<td>-10.1029</td>
</tr>
<tr>
<td>$f_{21}$</td>
<td>-10.5364</td>
</tr>
</tbody>
</table>

Notes: Notation ‘+’ and ‘≈’ indicates that the enhanced algorithm provided better, comparable or worse results than DE/rand/1/bin. In the cases denoted by ‘‡’ both algorithms achieved the global minimum and the standard deviation equals to 0.

For this reason, we mention the average number of the cost function evaluation.
Table 1  Experimental results for the benchmark set 1 (continued)

<table>
<thead>
<tr>
<th>Funct.</th>
<th>Mean</th>
<th>Stdev</th>
<th>Mean</th>
<th>Stdev</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1$</td>
<td>2.81E-18</td>
<td>2.14E-18</td>
<td>2.26E-20</td>
<td>2.20E-20</td>
</tr>
<tr>
<td>$f_2$</td>
<td>1.56E-15</td>
<td>5.81E-16</td>
<td>7.86E-17</td>
<td>3.12E-17</td>
</tr>
<tr>
<td>$f_3$</td>
<td>1.86E-39</td>
<td>1.20E-39</td>
<td>6.66E-43</td>
<td>3.00E-43</td>
</tr>
<tr>
<td>$f_4$</td>
<td>1.51E-11</td>
<td>8.35E-12</td>
<td>3.84E-14</td>
<td>1.45E-13</td>
</tr>
<tr>
<td>$f_5$</td>
<td>7.96E-02</td>
<td>2.84E-01</td>
<td>9.99E-02</td>
<td>5.58E-01</td>
</tr>
<tr>
<td>$f_6$</td>
<td>25.400</td>
<td>23.828</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f_7$</td>
<td>5.19E-03</td>
<td>1.31E-03</td>
<td>3.72E-03</td>
<td>8.56E-04</td>
</tr>
<tr>
<td>$f_8$</td>
<td>172.348</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f_9$</td>
<td>0.00E+00</td>
<td>0.00E+00</td>
<td>1.99E-02</td>
<td>3.99E-02</td>
</tr>
<tr>
<td>$f_{10}$</td>
<td>3.68E-10</td>
<td>1.34E-10</td>
<td>3.15E-11</td>
<td>9.35E-12</td>
</tr>
<tr>
<td>$f_{11}$</td>
<td>139.622</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f_{12}$</td>
<td>2.24E-20</td>
<td>1.57E-17</td>
<td>1.29E-21</td>
<td>9.27E-22</td>
</tr>
<tr>
<td>$f_{13}$</td>
<td>2.00E-18</td>
<td>1.62E-18</td>
<td>1.99E-20</td>
<td>3.87E-20</td>
</tr>
<tr>
<td>$f_{14}$</td>
<td>0.9980038</td>
<td>1.65E-10</td>
<td>0.9980038</td>
<td>2.49E-10</td>
</tr>
<tr>
<td>$f_{15}$</td>
<td>4.78E-05</td>
<td>9.26E-06</td>
<td>4.78E-05</td>
<td>9.26E-06</td>
</tr>
<tr>
<td>$f_{16}$</td>
<td>-1.0316285</td>
<td>5.45E-08</td>
<td>-1.0316285</td>
<td>2.11E-07</td>
</tr>
<tr>
<td>$f_{17}$</td>
<td>0.398</td>
<td>2.95E-09</td>
<td>0.398</td>
<td>2.34E-08</td>
</tr>
<tr>
<td>$f_{18}$</td>
<td>3.00E+00</td>
<td>1.85E-09</td>
<td>3.00E+00</td>
<td>4.82E-08</td>
</tr>
<tr>
<td>$f_{19}$</td>
<td>-9.8569</td>
<td>6.75E-01</td>
<td>-9.8626</td>
<td>5.98E-01</td>
</tr>
<tr>
<td>$f_{20}$</td>
<td>-10.4029</td>
<td>2.26E-01</td>
<td>-10.3217</td>
<td>2.37E-01</td>
</tr>
<tr>
<td>$f_{21}$</td>
<td>-10.4683</td>
<td>2.34E-01</td>
<td>-10.4246</td>
<td>2.18E-01</td>
</tr>
</tbody>
</table>

5.2 Experimental results of benchmark set 1

The comparison of the original versions of DE and their enhanced variants and the related experimental results of benchmark set 1 are shown in Table 1. The results are means and standard deviations achieved during 50 independent runs. To make a fair comparison, the Wilcoxon signed-rank test at the $\alpha = 0.05$ level has been conducted between the original DE algorithms (DE/rand/1/bin, jDE, CoDE) and their enhanced variants (DE/deg/l/bin, jDE/deg, CoDE/deg). If both compared algorithms achieve the global minimum and the standard deviation equals to 0.0, the mean number of the objective function evaluation will be mentioned. We consider that the algorithm is better if the numbers of the objective function evaluations is lower by at least 5%. Notation ‘+’, ‘≈’ and ‘–’ indicates that the enhanced algorithm of DE provided better, comparable or worse results than the original version of the algorithm.

The first novel algorithm – DE/deg/l/bin is based on the principle of the original version of DE/rand/1/bin. On the basis of the experimental results, the control parameters $F$ and $CR$ have been set as follows: $F = 0.5$ and $CR = 0.9$, respectively. The pseudo code of the DE/deg/l/bin algorithm is described in Algorithm 1. As we can see, in the case of the first three uni-modal functions $f_1$–$f_3$ the DE/deg/l/bin provided better results than the DE/rand/1/bin.

The better results have been also achieved for the step function $f_6$, quartic function $f_7$, multi-modal functions $f_{10}$–$f_{12}$ and $f_{13}$, and for the low-dimensional functions $f_{13}$ and $f_{15}$. For the fourth uni-modal function $f_4$ the DE/deg/l/bin provided worse results as well as for the low-dimensional test functions $f_{16}$ and $f_{18}$. For the rest of test functions from the benchmark set 1 the results are comparable. The DE/deg/l/bin provided better results in 14, comparable results in 4, and worse results in 3 cases from 21 test functions in the benchmark set 1.

In the case of the jDE algorithm, we have used the following settings: $\tau_1 = \tau_2 = 0.1$, $F = 0.1$, and $F_\alpha = 0.9$ as suggested by Brest et al. (2006). The enhanced jDE algorithm denoted as jDE/deg achieved better results in comparison with its original version for all uni-modal test functions $f_1$–$f_3$ the DE/deg/l/bin provided better results than the DE/rand/1/bin. The better results have been achieved for the step function $f_6$, quartic function $f_7$, multi-modal functions $f_{10}$–$f_{12}$, and $f_{13}$, and for the low-dimensional functions $f_{13}$ and $f_{15}$. For the fourth uni-modal function $f_4$ the DE/deg/l/bin provided worse results as well as for the low-dimensional test functions $f_{16}$ and $f_{18}$. For the rest of test functions from the benchmark set 1 the results are comparable. The DE/deg/l/bin provided better results in 14, comparable results in 4, and worse results in five cases from 21 test functions in benchmark set 1. The algorithm jDE/deg in comparison with its original version has been the less successful in the case of the low-dimensional test functions.

The composite DE algorithm (CoDE) uses three trial vector generation strategies (DE/rand/1/bin, DE/rand/2/bin, and DE/current-to-rand) and three control parameter settings ($[F = 1.0, \ CR = 0.1]$, $[F = 1.0, \ CR = 0.9]$, and $[F = 0.8, \ CR = 0.2]$). These strategies and settings are randomly combined to generate trial vectors. In this work, we have changed the pool of the parameters settings such that $[F = 0.9, \ CR = 0.1]$, $[F = 0.9, \ CR = 0.9]$, and
[\(F = 0.8, \ CR = 0.2\)] because the CoDE and its enhanced version CoDE/deg achieved better results with this setting in several cases. The population size has been set as well as in the previous cases to \(NP = 100\).

**Figure 3** The convergence rate for the selected DE algorithms and their enhanced versions, (a) function \(f_i\) from the benchmark set 1 (b) function \(f_i\) from the benchmark set 1 (c) function \(f_{10}\) from the benchmark set 1 (see online version for colours)

Note: The horizontal axis depicts the number of cost function evaluations, the vertical axis represents the mean of function values over 50 independent runs.

In the benchmark set 1, CoDE/deg provided better results in 15 cases from 21 test functions, more precisely in all uni-modal functions \(f_1-f_6\), step function \(f_6\), quartic function \(f_7\), multi-modal functions \(f_{10}-f_{13}\), and low-dimensional functions \(f_{15}\), and \(f_{18}-f_{21}\). CoDE/deg achieved the worse results in one test function, namely \(f_8\). For the rest of test functions, the results are comparable.

In Figures 3(a), 3(b), and 3(c), the convergence rates of all algorithms for the uni-modal test function \(f_7\), quartic function \(f_7\), and multi-modal test function \(f_{10}\) from the benchmark set 1 are depicted.

### 5.3 Experimental results of benchmark set 2

The comparison of the original version of DE/rand/1/bin and the DE/deg/1/bin, original version of jDE and jDE/deg, and the original version of CoDE and the CoDE/deg, and the related experimental results of benchmark set 2 are shown in Table 2. As well as in the previous case, the settings of the control parameters for the algorithm DE/deg/1/bin achieved better results for three uni-modal test functions \(f_2, f_4, \) and \(f_5\). Better results have been also achieved for the multi-modal test functions \(f_8, f_{11}-f_{15}, \) and \(f_{17}-f_{19}\), and for the composition test functions \(f_{21}-f_{23}, f_{26}, \) and \(f_{28}\). On the other hand, the DE/deg/1/bin provided the worse results in the case of the uni-modal test function \(f_7\). Next, the worse results have been achieved for the multi-modal test functions \(f_{24}, f_{25}\), and for the composition functions \(f_{24}\) and \(f_{27}\). For the rest of test functions, the comparable results have been provided.

The settings of the control parameters for the algorithm jDE and its enhanced version is the same as in the case of the benchmark set 1. The jDE/deg provided better results for four uni-modal test functions \(f_1, f_2, f_4, \) and \(f_5\). Next, the better results have been achieved for the multi-modal functions \(f_8, f_{11}-f_{15}, f_{17}, f_{18}, \) and \(f_{20}\), and for composition functions \(f_{22}, f_{23}, f_{25}, \) and \(f_{26}\). The worse results have been achieved for the multi-modal functions \(f_{24}, f_{16}\), and for the composition functions \(f_{24}\) and \(f_{27}\). For the rest of the test functions from the benchmark set 2, the results are comparable. The jDE/deg provided better results in 17 from 28 test functions from the benchmark set 2, in six cases the results are comparable and for five test functions the jDE/deg achieved worse results.

As well as in the previous case, the settings of the control parameters is the same as in the case of the benchmark set 1 for CoDE and its enhanced version CoDE/deg. The CoDE/deg provided better results for all uni-modal test functions \(f_1-f_6\). The better results have been also achieved for multi-modal functions \(f_8, f_{11}, f_{12}, f_{13}, f_{15}, f_{17}, \) and \(f_{18}\), and for composition functions \(f_{21}, f_{24}, f_{26}-f_{28}\). CoDE/deg provided worse results in two cases of the multi-modal functions \(f_{17}\) and \(f_{19}\), and in the one case of the composition functions \(f_{21}\). For the rest of test functions, the results are comparable.
<table>
<thead>
<tr>
<th>Function</th>
<th>Mean</th>
<th>Stdev</th>
<th>Mean</th>
<th>Stdev</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1$</td>
<td>9.21E-0</td>
<td>4.48E-29</td>
<td>1.11E-29</td>
<td>4.08E-29</td>
</tr>
<tr>
<td>$f_2$</td>
<td>4.69E+05</td>
<td>3.67E+05</td>
<td>3.30E+05</td>
<td>1.99E+05</td>
</tr>
<tr>
<td>$f_3$</td>
<td>8.25E-01</td>
<td>3.48E+00</td>
<td>1.29E+03</td>
<td>5.11E+03</td>
</tr>
<tr>
<td>$f_4$</td>
<td>1.54E+03</td>
<td>5.27E+02</td>
<td>8.17E+02</td>
<td>3.93E+02</td>
</tr>
<tr>
<td>$f_5$</td>
<td>1.58E-27</td>
<td>4.32E-27</td>
<td>1.42E-16</td>
<td>9.95E-16</td>
</tr>
<tr>
<td>$f_6$</td>
<td>9.14E+04</td>
<td>2.89E+01</td>
<td>1.58E+01</td>
<td>1.34E+01</td>
</tr>
<tr>
<td>$f_7$</td>
<td>8.92E-02</td>
<td>1.29E+01</td>
<td>3.26E+01</td>
<td>1.34E+01</td>
</tr>
<tr>
<td>$f_8$</td>
<td>2.09E+01</td>
<td>4.91E-02</td>
<td>2.09E+01</td>
<td>4.89E-02</td>
</tr>
<tr>
<td>$f_9$</td>
<td>3.85E+00</td>
<td>2.19E+00</td>
<td>1.02E+01</td>
<td>3.85E+00</td>
</tr>
<tr>
<td>$f_{10}$</td>
<td>6.31E-03</td>
<td>0.00E+00</td>
<td>1.58E+01</td>
<td>1.34E+01</td>
</tr>
<tr>
<td>$f_{11}$</td>
<td>2.00E+02</td>
<td>1.89E-01</td>
<td>1.18E+01</td>
<td>1.99E-02</td>
</tr>
<tr>
<td>$f_{12}$</td>
<td>9.14E+00</td>
<td>2.89E+00</td>
<td>1.58E+01</td>
<td>1.34E+01</td>
</tr>
<tr>
<td>$f_{13}$</td>
<td>8.92E-02</td>
<td>1.29E+01</td>
<td>3.26E+01</td>
<td>1.34E+01</td>
</tr>
<tr>
<td>$f_{14}$</td>
<td>2.09E+01</td>
<td>4.91E-02</td>
<td>2.09E+01</td>
<td>4.89E-02</td>
</tr>
<tr>
<td>$f_{15}$</td>
<td>2.00E+02</td>
<td>1.89E-01</td>
<td>1.18E+01</td>
<td>1.99E-02</td>
</tr>
<tr>
<td>$f_{16}$</td>
<td>9.14E+00</td>
<td>2.89E+00</td>
<td>1.58E+01</td>
<td>1.34E+01</td>
</tr>
<tr>
<td>$f_{17}$</td>
<td>8.92E-02</td>
<td>1.29E+01</td>
<td>3.26E+01</td>
<td>1.34E+01</td>
</tr>
<tr>
<td>$f_{18}$</td>
<td>2.09E+01</td>
<td>4.91E-02</td>
<td>2.09E+01</td>
<td>4.89E-02</td>
</tr>
</tbody>
</table>

Notes: Notation ‘+’, ‘≈’, and ‘−’ indicates that the enhanced algorithm provided better, comparable or worse results than DE/rand/1/bin. In the cases denoted by † both algorithms achieved the global minimum and the standard deviation equals to 0. For this reason, we mention the average number of the cost function evaluation. For the test function $f_{28}$ both algorithms achieved the mean error 3.00E+02 and the std. dev. was 0.00E+00.
Table 2  Experimental results for the benchmark set 2  
(continued)

<table>
<thead>
<tr>
<th>Funct.</th>
<th>Mean</th>
<th>St dev</th>
<th>Mean</th>
<th>Stdev</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_{20}$</td>
<td>1.15E+01</td>
<td>4.55E-01</td>
<td>1.09E+01</td>
<td>6.40E-01</td>
</tr>
<tr>
<td>$f_{21}$</td>
<td>3.34E+02</td>
<td>5.57E+01</td>
<td>3.42E+02</td>
<td>4.93E+01</td>
</tr>
<tr>
<td>$f_{22}$</td>
<td>3.69E+02</td>
<td>1.10E+02</td>
<td>4.88E+02</td>
<td>1.81E+02</td>
</tr>
<tr>
<td>$f_{23}$</td>
<td>5.02E+03</td>
<td>8.08E+02</td>
<td>4.17E+03</td>
<td>6.56E+02</td>
</tr>
<tr>
<td>$f_{24}$</td>
<td>2.17E+02</td>
<td>1.34E+01</td>
<td>2.02E+02</td>
<td>1.84E+00</td>
</tr>
<tr>
<td>$f_{25}$</td>
<td>6.99E+00</td>
<td>2.61E+02</td>
<td>1.40E+01</td>
<td>5.33E-03</td>
</tr>
<tr>
<td>$f_{26}$</td>
<td>9.39E-03</td>
<td>2.00E+02</td>
<td>5.33E-03</td>
<td>1.02E+02</td>
</tr>
<tr>
<td>$f_{27}$</td>
<td>6.03E+02</td>
<td>1.60E+02</td>
<td>3.95E+02</td>
<td>1.02E+02</td>
</tr>
<tr>
<td>$f_{28}$</td>
<td>460,872</td>
<td>244,218</td>
<td>+</td>
<td></td>
</tr>
</tbody>
</table>

Notes: Notation ‘+’, ‘≈’, and ‘–’ indicates that the enhanced algorithm provided better, comparable or worse results than DE/rand/1/bin. In the cases denoted by † both algorithms achieved the global minimum and the standard deviation equals to 0. For this reason, we mention the average number of the cost function evaluation. For the test function $f_{28}$ both algorithms achieved the mean error 3.00E+02 and the std. dev. was 0.00E+00.

In Figures 4(a), 4(b), and 4(c), the convergence rates of all algorithms for the uni-modal test function $f_2$, and multimodal test functions $f_{13}$ and $f_{15}$ from the benchmark set 2 are depicted.

5.4 Comparison of the enhanced algorithms

In this section, the comparison of the enhanced algorithms – DE/deg/1/bin, jDE/deg and CoDE/deg is described. To make a fair comparison, Friedman rank test with post hoc testings has been applied.

The CoDE/deg algorithm achieved the best results from all enhanced algorithms for all uni-modal test function $f_1$–$f_5$, quartic function $f_7$, multi-modal test functions $f_{10}$, $f_{12}$, and $f_{13}$, and for all low-dimensional test functions $f_{14}$, $f_{15}$, and $f_{19}$–$f_{21}$ from the benchmark set 1. The algorithm denoted as jDE/deg achieved the best results for the multi-modal test functions $f_6$, $f_9$, and $f_{13}$. The enhanced algorithm DE/deg/1/bin achieved the best results in the case of the Step function $f_6$.

Unlike the benchmark set 1, in the case of the benchmark set 2, for the first view it has not been so easy to say, which algorithm has provided the best results for the most of testing functions. The DE/deg/1/bin achieved the best results for the multi-modal test functions $f_1$, $f_6$, $f_{12}$, and $f_{13}$, and for the composition function $f_{27}$. The algorithm jDE/deg achieved the highest convergence rate for two uni-modal test functions $f_1$ and $f_6$, for multi-modal test functions $f_{10}$, and for the composition functions $f_{19}$, and $f_{19}$, and for the composition functions $f_{21}$–$f_{23}$, and $f_{28}$. Finally, the CoDE/deg algorithm achieved the highest convergence rate for three uni-modal test functions $f_2$–$f_5$, multi-modal test functions $f_6$ and $f_{13}$, and for the composition functions $f_{24}$ and $f_{26}$.

6 Conclusions

Visualisation of the evolutionary algorithms dynamics using complex networks has been described in some publications from the recent years (see Zelinka et al., 2010, 2012; Davendra et al., 2014). These publications deal with the
investigation of the relationships between the evolutionary algorithms dynamics and the CNs creation and with the analysis of the properties of such created complex networks. However, we missed the research regarding the incorporation of the analysed properties of the CN back into the evolutionary algorithms to improve their performance.

The motivation of this work is to model the DE dynamics using the CNs to better understand the relationships between the individuals in the population during generations and use this knowledge to improve the DE performance. We have introduced the approach of the parent selection in the mutation step which is based on the usefulness of the individual in the population. This usefulness is represented by the new property ‘strength’ added to the individual that corresponds with the strength of the node representing this individual in the CN. This approach has been incorporated into three state-of-the-art DE algorithms, DE/rand/1/bin, jDE, and CoDE and three novel algorithms have been developed – DE/deg/1/bin, jDE/deg and CoDE/deg. The performance of the enhanced algorithms is evaluated on two well-known benchmarks [Yao et al. (1999) and CEC2013 (Liang et al., 2013a)] and according to the results mentioned in Tables 1 and 2 a better performance is obtained compared with the original versions of the DE variants.

The novel algorithm DE/deg/1/bin provided better results in 14 from 21 test functions in the benchmark set 1 (in four cases the results are comparable and in three cases the DE/deg/1/bin achieved worse results) and in 17 from 28 test functions in the benchmark set 2 (for four test function the results are comparable and in seven cases the DE/deg/1/bin achieved the worse results) than its original version DE/rand/1/bin.

The jDE/deg achieved better results in 12 from 21 test functions in the benchmark set 1 (for four test functions the results are comparable and in five cases the jDE/deg provided worse results) and in 17 from 28 test functions in the benchmark set 2 (for six test functions the results are comparable and in five cases the jDE/deg achieved worse results) than its original version jDE.

Finally, the CoDE/deg provided better results in 15 from 21 test functions in the benchmark set 1 (for five test functions the results are comparable and in one case the CoDE/deg achieved worse results) and in 17 from 28 test functions in the benchmark set 2 (in eight cases the results are comparable and for three test functions CoDE/deg achieved worse results) than its original version CoDE.

In Subsection 5.4, the comparison of the enhanced algorithms is described, where the enhanced algorithms are compared with each other. To make a fair comparison Friedman rank test with post hoc testings has been used. For the benchmark set 1, the CoDE/deg achieved the best results in the most of the test functions, specifically in 14 from 21 test functions (the jDE/deg algorithm provided the best results in three cases and DE/deg/1/bin in 1 case from 21 test functions). In the case of the benchmark set 2, the jDE/deg algorithm achieved the best results for nine test functions, the CoDE/deg provided the best results for seven test functions, and the DE/deg/1/bin achieved the best results for 6 from 28 test functions of the benchmark set 2.

In the future, we would like to extend our study by other properties (closeness centrality, betweenness centrality, etc.) of the corresponding CNs which could be incorporated into the DE to improve its convergence rate.

Acknowledgements

The following grants are acknowledged for the financial support provided for this research: Grant Agency of the Czech Republic – GACR P103/15/06700S and VSB-TU internal grant SGS 2016/175.

References


