ISSN: 1803-3814

DYNAMIC OF FIREWORK ALGORITHM ANALYZED WITH COMPLEX NETWORK

Tomas Kadavy, Michal Pluhacek, Adam Viktorin and Roman Senkerik

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

Abstract: *In this paper, a visualization of Firework Algorithm (FWA) inner dynamics as an evolving complex network is presented. Recent research in unconventional controlling and simulation of metaheuristic dynamics shows that this kind of visualization technique has been utilized only for algorithms with some social communication or behavior leading to sharing information across the population. However, provided analysis suggests that the network can identify some types of surface of tested functions.*

Keywords: *Firework Algorithm, FWA, Complex Network, Surface Analysis*

1 Introduction

In this paper, we are presenting a visualization of Firework Algorithm (FWA) inner dynamics as an evolving complex network. The FWA is an algorithm for numerical optimization, which has been introduced in 2010 by authors Tan and Zhu [1] with promising results. Since then, many new versions and applications of this algorithm appeared [2]. The algorithm is based on fireworks explosions in the sky. This algorithm has similar characteristics like some scatter search algorithms [3]. This FWA can be described as non-bio-inspired algorithms like a water drop algorithm [4] or brainstorm optimization [5].

In this paper, the dynamic of FWA is transformed into the evolving complex network [6]. The population is visualized as an evolving complex network that usually exhibits non-trivial features – e.g. degree distribution, clustering, centralities and in between. 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. This complex networks can be more analyzed using different techniques [7, 8]. Typically, this analysis is made on Swarm Intelligence (SI) algorithms [9-12] which have some social behavior or communication across particles. This social behavior can be transferred into the complex network. Can be similar behavior or communication observed using the complex network on FWA? Our simulation experiment presents the original approach for analyzing the complex dynamics of an algorithm based not on social behavior, but mostly on random/local search engines. Also, can this network be used to observe and identify the surface of tested function?

The paper is structured as follows. The FWA is described in Section 2. Network Design is described in next section, Section 3. Section 4 contains a description of used test functions. The experiment setup is detailed in Section 5. Results and conclusion follow in sections 6 and 7.

2 Firework Algorithm

This section describes the FWA algorithm like it was proposed by Tan and Zhu [1]. The FWA is an algorithm that is inspired by fireworks explosion in a night sky. This algorithm is initialized with a random population of fireworks $\boldsymbol{X}$. The x_i firework position is represented as coordinates in n-dimensional space of solutions. These coordinates are parameters of the optimized problem. The number of the fireworks is defined by parameter NP; this parameter is set by the user. Moreover, the user defines the parameters like number of iterations of the algorithm (terminal condition), Gaussian mutation $\widehat{m}$, number of sparks m, parameters a and b and constant $\hat{A}$. This algorithm consists of four parts: explosion operator, mutation operator, mapping rule and selection strategy. These parts and adjustable parameters are more explained in next sub-sections.

The realization of FWA is as follows:

1. Randomly generate NP fireworks in the n-dimensional search space.
2. Obtain the fitness values of these generated fireworks by fitness function.
3. Calculate the number of generated sparks and their amplitude for each firework by explosion operator.
4. Use Gaussian mutation to generate new random sparks by mutation operator.
5. Apply mapping rule to all generated sparks.

6. Calculate fitness values of sparks and by applying selection strategy pick the selected sparks as new fireworks.

7. If the terminal conditions are met, stop the algorithm. Otherwise, continue the iteration process from 3.

There can be more or different terminal conditions defined by the user. For example, a total number of fitness evaluations (*FE*) instead of a number of iterations of the algorithm.

2.1 Initialization

The initial *NP*, the number of fireworks, fireworks $\boldsymbol{X}$ are randomly generated with uniform distribution from the range which is specified for the problem by lower and upper bounds defined by the optimized problem with dimensionality *dim*.

In the initialization phase, the adjustable parameters mentioned before has to be defined as well.

2.2 Explosion Operator

The number of sparks generated from each firework is determined by the firework fitness value. The firework with better fitness value produces more sparks (the lower cost function *f(x)*, the better fitness value). This number of sparks is calculated by explosion strength in (1).

$$S_i = m \cdot \frac{Y_{max} - f(x_i) + \varepsilon}{\sum_{i=1}^{NP}[Y_{max} - f(x_i)] + \varepsilon}. \tag{1}$$

where S_i is the number of sparks for firework i, m is the total number of sparks defined by the user. Y_{max} means the fitness value of the worst individual (firework). Function $f(x_i)$ is the fitness value for the individual firework i. The last parameter ε is used to prevent the denominator from becoming zero and it should be the smallest possible number.
There is also a limitation of the number of generated spark defined as (2).

$$\hat{S}_i = \begin{cases} round(a \cdot m), & if\ s_i < a \cdot m \\ round(b \cdot m), & if\ s_i > b \cdot m. \\ round(s_i), & otherwise \end{cases} \tag{2}$$

where a and b are constants defined by the user (these constants has to be $a<b<1$), $\hat{s_i}$ is the limitation of the number of sparks and *round()* is the rounding function.

The amplitude for generated sparks is then calculated by explosion amplitude in (3). Like the previous, explosion strength, the amplitude of explosion is defined by firework fitness function. The better fitness value is, the smaller is the amplitude of explosion and vice versa.

$$A_i = \hat{A} \cdot \frac{f(x_i) - Y_{min} + \varepsilon}{\sum_{i=1}^{NP}[f(x_i - Y_{min})] + \varepsilon}. \tag{3}$$

where A_i is the amplitude of i firework. $\hat{A}$ is a constant defined by the user and means the sum of all amplitudes. Y_{min} means the fitness value of the best firework.

The new sparks are generated in randomly chosen dimensions $\boldsymbol{z}$ and the position is calculated in (4).

$$\hat{x}_j^k = x_i^k + U(-A_i, A_i). \tag{4}$$

where $\hat{x}_j^k$ is spark j in dimension k ($k \in \boldsymbol{z}$) generated from firework x_i. U is a random number from a uniform distribution in the range of the explosion amplitude of i firework.

2.3 Mutation Operator

To maintain the diversity of the population, some mutation operator is needed. For FWA, the Gaussian mutation is used. The sparks are generated as follows:

1. Choose random firework i.
2. Compute new spark using formula (5).

3. If the number of generated spark by Gaussian mutation reaches the value $\hat{m}$, stop generating next sparks

$$\hat{x}_j^k = x_i^k \cdot N(1,1) \tag{5}$$

where $\hat{x}_j^k$ is spark j in dimension k ($k \in \boldsymbol{z}$) generated from firework x_i. Vector $\boldsymbol{z}$ are randomly chosen dimensions like in section Explosion Operator. N is a random number from normal (Gaussian) distribution with mean 1 and variance 1.

2.4 Mapping Rule

This rule ensures, that all previously generated sparks are in feasible space. If any spark lies outside of the available search space, its mapped back to allowed space. This mapping rule defined as (6).

$$\hat{x}_i^k = B_L^k + \hat{x}_i^k mod(B_U^k - B_L^k) \tag{6}$$

where $\hat{x}_i^k$ is i particle in k dimension, B_L^k and B_U^k are lower and upper boundaries of the available search space in k dimension. The *mod* represents modular operation.

2.5 Selection Strategy

Some of the generated sparks need to be selected and passed into the new iteration. These selected sparks will become new fireworks. For this selection, the distance-based strategy is used to maintain the diversity of the population. The spark that is farther from the others has the greater chance to be selected than those sparks near the other sparks. The first chosen spark is always the one with the best fitness value. Others (*NP*-1) individuals are chosen by roulette method. The possibility of choosing the spark into next iteration is calculated in (7).

$$p_i = \frac{R_i}{\sum_{j=1}^{K} R_j} \tag{7}$$

where p_i is the possibility of the i spark, R_i is the sum of distances of the i spark, K is the number of all generated sparks. The Euclidean distance is used to compute the R_i in formula (8).

$$R_i = \sum_{j=1}^{K} d(\hat{x}_i, \hat{x}_j) = \sum_{j=1}^{K} \|\hat{x}_i - \hat{x}_j\| \tag{8}$$

where K is the number of all sparks, $\hat{x}_i$is the spark for which the R_i is computed and $\hat{x}_j$ are others sparks where $j \in K$.

The whole FWA is depicted in the pseudo-code below.

```
Algorithm pseudo-code 1: FWA
1.  Randomly initialize NP fireworks
2.  while terminal condition not met
3.    count fireworks fitness values
4.    for i = 1 to NP do
5.      calculate Si
6.      calculate Ai
7.      generate sparks of i firework
8.    end
9.    for j = 1 to m̂ do
10.     Gaussian mutation
11.   end
12.   apply mapping rule
13.   selection strategy for new fireworks
14.  end
```

3 Network Design

The network is created as a history of contributions. Despite the fact that the inner dynamic of evolutionary algorithms has been transferred to the complex network multiple times [9-12], the FWA and its unique communication scheme required a development of a new technique, which is proposed in this section. In each iteration, there are *NP* fireworks. These fireworks create *K* sparks. Some of these sparks are transferred into a new iteration as new fireworks. Fireworks are then represented as the nodes in the network. These nodes are labeled 1…*NP* for each iteration. The nodes (fireworks) are sorted by their fitness values before labeling so that the best node (smallest fitness value) gets number 1 and the worst node gets number *NP*. The edge between nodes represents spark that creates a new firework in next iteration. The initial node of the edge represents the firework from which the spark is created. The terminal node is the firework in the next iteration created by the spark. With that rule, the initial node from *t* iteration can have from 0 to *NP* edges and terminal node can only have one edge as input.

An example of the network with five fireworks in four iterations is shown in Figure 1. Blue edges indicate the spark with the best fitness function value. The blue edge direction can only be towards the node number one. The first iteration is on top of the figure, and the last iteration is on the bottom. From the first iteration, only sparks from last three fireworks created new ones (fireworks 3, 4 and 5) in the second iteration. From the second iteration to last, firework labeled as 1 (with the best fitness value), contribute to improving its solution.

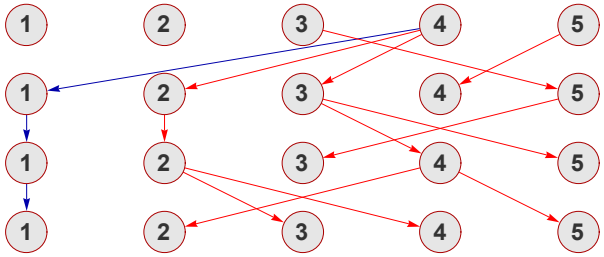

Figure 1: The example of FWA network.

4 Test Functions

For the simulation experiment, a set of 5 classic functions were selected. The set consists of unimodal and multimodal functions:

- Sphere function (f_1) (9),
- Rosenbrock function (f_2) (10),
- Rastrigin function (f_3) (11),
- Schwefel function (f_4) (12),
- Egg holder function (f_5) (13).

$$f(x)_1 = \sum_{i=1}^{dim} x_i^2, \tag{9}$$

$$f(x)_2 = \sum_{i=1}^{dim-1} [100 \cdot (x_{i+1} - x_i^2)^2 + (1 - x_i)^2], \tag{10}$$

$$f(x)_3 = 10 \cdot dim + \sum_{i=1}^{dim} [x_i^2 - 10 \cdot cos(2\pi x_i)], \tag{11}$$

$$f(x)_4 = \sum_{i=1}^{dim} [-x_i \cdot sin(|x_i|^{0.5})], \tag{12}$$

$$f(x)_5 = \sum_{i=1}^{dim-1} \left[-(x_{i+1} + 47) \cdot sin\left(\sqrt{\left| x_{i+1} + \frac{x_i}{2} + 47 \right|} \right) - x_i \cdot sin\left(\sqrt{|x_i - (x_{i+1} + 47)|} \right) \right]. \tag{13}$$

5 Experimental Setup

The experiments were performed for test functions dimensions 2, 10 and 15. The number of iterations was set to 50. The control FWA parameters were set accordingly to [1]. The number of fireworks, population size (*NP*), was set to 5 for all dimensions. The number of sparks (m) was set as 50. Parameters a and b were set as 0.8 and 0.04. Other constant settings were following: $\hat{A} = 40$ and $\hat{m} = 5$.

The basic logical assumption was that the *longest path of steady improvement* in the network (i.e. the path between nodes labeled 1 and joined with blue edges) would be observable mostly for the unimodal function (e.g. f_1).

6 Results

The results for the aforementioned longest paths of the stable improvement are given in Table 1. In this table, the mean, minimal, maximal and standard deviation are presented. Every test on function and dimension were performed 11 times.

Results depicted in Table 1 confirm the anticipated logical assumption made in the previous section. For the unimodal functions (f_1 and f_2), the observed path is quite longer compared to the results for the multimodal functions. The differences are decreasing with the higher dimension setting. These trends are graphically confirmed also in Figures 2 – 3 for 40 iterations. For unimodal functions, the path of steady improvement seems to be present more often at the end of the recorded optimization process. For multimodal functions, these path appears to be scattered across whole iteration process.

Table 1: Longest paths of steady improvements in the network

Function	Dimension					
	2		***10***		***15***	
f_1	$Avg.$	16.82	$Avg.$	13.27	$Avg.$	11.46
	Min	6	Min	8	Min	6
	Max	31	Max	18	Max	17
	$Std.Dev.$	8.04	$Std.Dev.$	3.72	$Std.Dev.$	4.32
f_2	$Avg.$	12.64	$Avg.$	11.73	$Avg.$	10.73
	Min	7	Min	4	Min	6
	Max	18	Max	25	Max	20
	$Std.Dev.$	3.41	$Std.Dev.$	5.27	$Std.Dev.$	3.88
f_3	$Avg.$	13.55	$Avg.$	10.00	$Avg.$	12.00
	Min	7	Min	6	Min	7
	Max	26	Max	14	Max	19
	$Std.Dev.$	6.15	$Std.Dev.$	2.86	$Std.Dev.$	4.29
f_4	$Avg.$	9.73	$Avg.$	5.91	$Avg.$	6.64
	Min	5	Min	4	Min	4
	Max	20	Max	8	Max	12
	$Std.Dev.$	5.57	$Std.Dev.$	1.64	$Std.Dev.$	2.50
f_5	$Avg.$	11.18	$Avg.$	6.82	$Avg.$	8.27
	Min	5	Min	4	Min	5
	Max	27	Max	8	Max	15
	$Std.Dev.$	6.62	$Std.Dev.$	1.40	$Std.Dev.$	3.35

On figure 2, for test function f_1, can be seen the presence of the *longest path of steady improvement*. For the lower dimension, the path is clearly longer among the higher dimension. On figure 3, on right side, the presence of the path of steady improvement on the early beginning can suggest that the FWA stuck in local minima due to the fact that at the ending the path shatters.

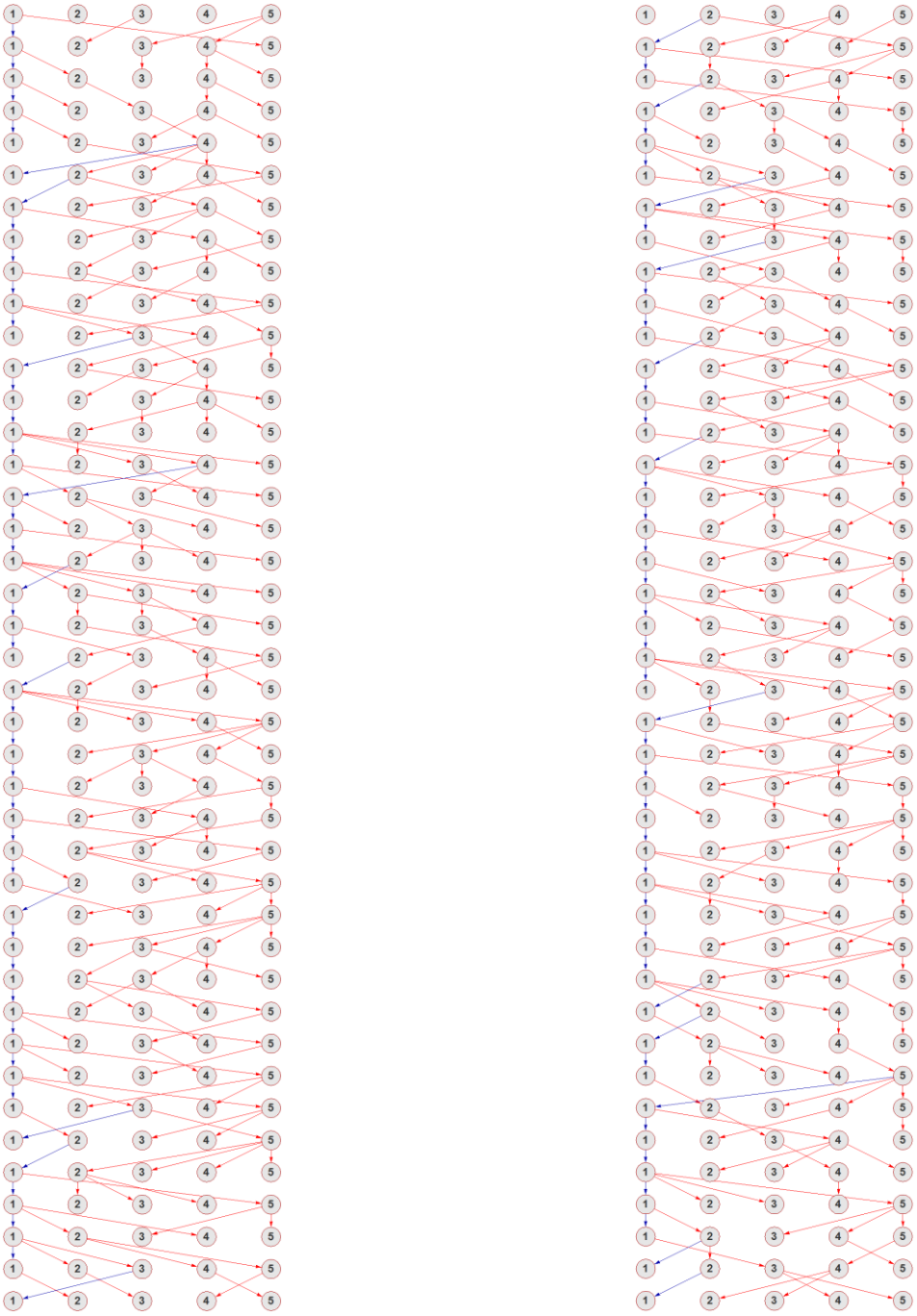

Figure 2: Evolving network of f_1 for dim = 2 (left figure) and dim = 15 (right figure).

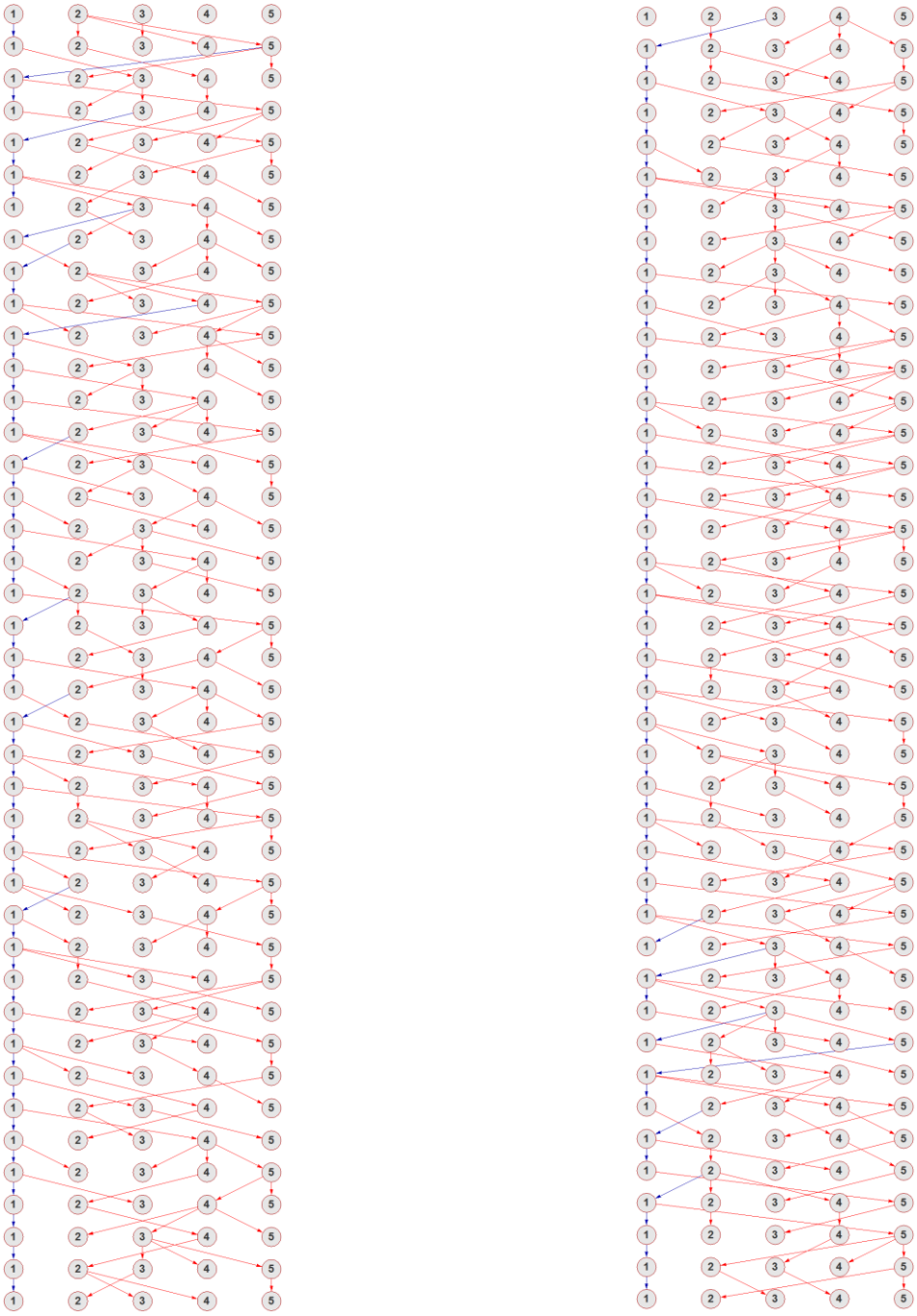

Figure 3: Evolving network of f_3 for dim = 2 (left figure) and dim = 10 (right figure)

7 Conclusion

In this paper, the possibility of simulation and simple analysis of complex network evolvement for firework algorithm inner dynamics is present. Our novel approach was tested on the set of five simple classical benchmark functions which consist of basic unimodal and multimodal types.

The preliminary results lend weight to the argument that the ability of a network to identify the surface type of optimized function seems to be present. Nevertheless, more and detailed in-depth study is required to be performed in this field.

Another phenomenon has been discovered. The network seems to have a lack of any other usable information. The results of this simple simulation study will be further used in future research to suggest possible improvements to building complex networks for the family of algorithms based on the local/random search techniques without accessible direct social/communication interactions.

Acknowledgement: This work was supported by Grant Agency of the Czech Republic – GACR P103/15/06700S, further by the Ministry of Education, Youth and Sports of the Czech Republic within the National Sustainability Programme Project no. LO1303 (MSMT-7778/2014. Also 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/2017/004.

References

[1] Tan Y., Zhu Y. (2010) Fireworks Algorithm for Optimization. In: Tan Y., Shi Y., Tan K.C. (eds) Advances in Swarm Intelligence. ICSI 2010. Lecture Notes in Computer Science, vol 6145. Springer, Berlin, Heidelberg

[2] S. Zheng, A. Janecek and Y. Tan. Enhanced Fireworks Algorithm. In: 2013 IEEE Congress on Evolutionary Computation. IEEE, 2013, pp. 2069-2077.

[3] Laguna M., Marti R. Scatter search: methodology and implementations in C. Boston: Kluwer Academic Publishers, c2003. ISBN 9781402073762.

[4] Shah-Hosseini H., the intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm. Int. J. Bio-Inspir. Comput. 1(1), 71-79 (2009)

[5] Shi Y., Brain storm optimization algorithm, in Advances in Swarm intelligence (Springer, Berlin, 2011), pp. 303-309

[6] Barrat, A., Barthelemy M., Vespignani A. Dynamical processes on complex networks. New York: Cambridge University Press, 2008. ISBN 9780521879507.

[7] Otte, Evelien; Rousseau, Ronald (2002). "Social network analysis: a powerful strategy, also for the information sciences". Journal of Information Science. 28 (6): 441–453

[8] Kudělka, M., Zehnalová, Š., Horák, Z., Krömer, P., & Snášel, V. (2015). Local dependency in networks. International Journal of Applied Mathematics and Computer Science, 25(2), 281-293.

[9] Pluhacek, M., Janostik, J., Senkerik, R., & Zelinka, I. (2016a). Converting PSO dynamics into complex network-Initial study. In T. Simos, & C. Tsitouras (Eds.), AIP Conference Proceedings (Vol. 1738, No. 1, p. 120021). AIP Publishing.

[10] Pluhacek, M., Senkerik, R., Janostik, J., Viktorin, A., & Zelinka, I. (2016b). Study on swarm dynamics converted into complex network. In Proceedings-30th European Conference on Modelling and Simulation, ECMS 2016. European Council for Modelling and Simulation (ECMS).

[11] Senkerik, R., Viktorin, A., Pluhacek, M., Janostik, J., & Davendra, D. (2016a). On the Influence of Different Randomization and Complex Network Analysis for Differential Evolution. In 2016 IEEE Congress on Evolutionary Computation (CEC) (pp. 3346-3353). IEEE.

[12] Senkerik, R., Viktorin, A., Pluhacek, M., Janostik, J., & Oplatkova, Z. K. (2016b). Study on the Time Development of Complex Network for Metaheuristic. In Artificial Intelligence Perspectives in Intelligent Systems (pp. 525-533). Springer International Publishing.