International Journal of Computational and Experimental Science and ENgineering (IJCESEN)

Vol. 9-No.2 (2023) pp. 91-101
http://dergipark.org.tr/en/pub/ijcesen

ISSN: 2149-9144

Research Article

PTGNG: An Evolutionary Approach for Parameter Optimization in the Growing Neural Gas Algorithm

Mohanad ALALKAWI[1], Shadi AL SHEHABI[2]*, Meltem Y. IMAMOGLU[3]

[1]Directorate General of Education in Diyala Govgovernorate, 32001, Diyala-Iraq
Email: mohanadalalkawi@gmail.com - **ORCID:** 0000-0002-1181-3053

[2]University of Turkish Aeronautical Association, Faculty of Engineering, Departmant of Computer Engineering, 06790, Ankara-Turkey
* **Corresponding Author** : **Email:** salshehabi@thk.edu.tr - **ORCID:** 0000-0003-0545-9104

[3]University of Turkish Aeronautical Association, Faculty of Engineering, Departmant of Computer Engineering, 06790, Ankara- Turkey
Email: meltemyi@gmail.com - **ORCID:** 0000-0002-8574-4097

Article Info:

DOI: 10.22399/ijcesen.1282146
Received : 12 April 2023
Accepted : 07 June 2023

Keywords

Growing Neural Gas
Parameter tuning
Evolutionary algorithm

Abstract:

Growing Neural Gas (GNG) algorithm is an unsupervised learning algorithm which belongs to the competitive learning family. Since then, GNG has been a subject to vaious developments and implementations found in the literatures for two main reasons: first, the number of neurons (i.e., nodes) is adaptive. Meaning, it is periodically changed through adding new neurons and removing old neurons accordingly in order to find the best network which captures the topological structure of the given data, and to reduce the overall error in that representation. Second, GNG algorithm has no restrictions when compared to other competitive learning algorithms, as it is both free in the space and the number of the neurons. In this paper, we propose and implement an evolutionary based approach, namely PTGNG, to tune GNG algorithm parameters for dealing with data in multiple dimensional space, namely, 2D, 3D, and 4D. The idea basically relies on finding the optimum set of parameter values for any given problem to be solved using GNG algorithm. The evolutionary algorithm by its nature searches a vast space of applicable solutions and evaluates each solution individually. When we implemented our approach of parameters tuning, we can note that GNG captured datasets topological structure with a smaller number of neurons and with a better accuracy. It also showed that the same results appeared when working on datasets with three and four dimensions.

1. Introduction

The Growing Neural Gas (GNG) is an unsupervised learning neural network algorithm which was proposed by Bernd Fritzke [1]. It is one of the competitive learning algorithms such that their common aim is to represent any data distribution in multidimensional space using lower number of neurons (nodes) [2]. GNG is a popularization from the Growing Cell Structure (GCS) [3], and it is based on the Neural Gas (NG) algorithm [4].

There are two alternatives of GNG algorithm. The first one, Robust Growing Neural Gas algorithm (RGNG) which was proposed in 2004 to make the GNG algorithm more robust than the original algorithm by adding parameters to enhance the GNG algorithm and the optimal numbers of nodes (neurons) is determined dynamically during the execution time [5]. The second one, Merge Growing Neural Gas (MGNG) algorithm was proposed in 2009 and it is a combination between Merge Neural Gas (MNG) algorithm and GNG algorithm [6]. The above mentioned algorithms are based and derived from self-organizing map algorithm (SOM) which is an unsupervised algorithm that is able to represent high dimensional data in a fixed network dimensionality namely, two-dimensional grid [7]. The NG algorithm was proposed to find the best data representation that based on the feature vectors. The NG algorithm is able to represent data in any dimension such that the reference vectors (neurons) are adapted without any fixed topology and there are two variants of the NG algorithm. First one, Enhance Neural Gas Algorithm (ENG) was proposed in 2005

to make the NG algorithm more stable than the previous algorithm by sorting the position and inputs of NG algorithm parameters [8]. The second one, Merge Neural Gas Algorithm (MNG) is comparable to NG algorithm, but it is work by three steps (initialization, sampling and matching and the last one is adoption), In the MNG algorithm they merge second and third steps of the ENG algorithm in one step [9].

The main advantage of GNG as compared to the other competitive learning algorithms is that the number of neurons increases periodically according to the distribution of the input data. The mechanism of changing the number of neurons during the learning process overcomes the problem of finding an appropriate number of neurons generated for reducing the error in the data representations.

Unfortunately, GNG algorithm, is like the other competitive algorithms, such that it has a lot of parameters that should be set correctly at the beginning of the learning process. Setting these parameters for a given data distribution is usually done experimentally [10]. Inappropriate tuning of these parameters may lead to represent poorly the data and also may stuck in deleting and adding neurons during the learning process [11]. The result provided by GNG gives more appropriate representation of 2 dimensional data than SOM and NG due to the original tuning of GNG parameters for representing two dimensional data [2]

The tuning of GNG parameters is usually done experimentally for each given data distribution. Therefore, tuning GNG parameters is time consuming. However, several parameter tuning algorithms such as, Evolutionary Algorithm (EA) [12], Relevance Estimation and Value Calibration of Parameter (REVAC) [13], F-Race [14] and Design of Experiments (DoE) [15] can be used for optimizing GNG parameters. The idea of REVAC is to find an optimal vector of parameters by estimation the distribution of promising values on the domain of each parameter and creation vectors based on values from this distribution. In F-Race algorithm, combinations of predefined parameter are examined, and the ones that perform poorly are discarded as soon as any relevance emerges. Design of Experiments (DoE) tries to reduce the number of required experiments for an analysis, while reserving high quality results. Experiments are considered to have input variables and output variables. It aims to optimize the input variables by comparing and evaluating the quality of the output variables. Genetic algorithms are evolutionary algorithms (EA) that optimize functions by modeling biological processes, they are known to be very effective methods to solve combinatorial optimization problems [16]. In [17] it shows that GA is more consistent than REVAC and F-Race, it shows also that GA is much faster than DoE and gives better solution quality. Therefore, we propose and implement an evolutionary based approach, namely PTGNG, to tune the GNG parameters for dealing with data described in multiple dimensional space.

2. Literature Review

2.1 Growing Neural Gas (GNG)

This model was proposed in 1995 by Fritzke [1] as an unsupervised learning model. GNG also belongs to the competitive learning family and it is a popularization of the GCS algorithm and based on the NG algorithm. What make this algorithm special from the other models is that its number of neurons changes periodically according to the need of these node for representing the data distribution. Unfortunately, this model works well when two dimensional data is present but it doesn't work properly if multidimensional data is present. This is done because of the inappropriate tuning of the parameters for data described in more than two dimensions [11].

GNG algorithm behaviour is controlled by six main parameters, the maximum connection age (AGEMAX), Cycle interval between nodes insertions (λ), Error reduction factor upon insertion (α), Wining node adaption factor (ε_b), Wining node neighbour adaption factor (ε_n) and the Error reduction factor for each cycle (β) [2]. These parameters must be carefully tuned if an accurate result is required either by applying prior knowledge about the nature of the presented problem to be solved or by setting them empirically. However, tuning these parameters based on the nature of one particular problem doesn't necessarily ensure the optimum results. Also, the empirical tuning of these parameters values for each case can be both extensive and time consuming [10].

2.2 Related Studies

In [18], Ventocilla, Elio, et al. suggested two methods for improving GNG in order to visualize the cluster patterns in large-scale and high-dimensional datasets. The first is to provide more precise and relevant 2D visual representations of cluster patterns in high-dimensional datasets, by avoiding connections that generate high-dimensional graphs that result in overplotting and clutter. The second one reduces the execution times of the learning phase by modeling and merging seperate parts of a big dataset.

In [19], Mendes, Carlos Augusto Teixeira, et al. suggested a new algorithm called the Fast Growing

Neural Gas (FGNG) algorithm such that some modifications were done on some steps in the GNG algorithm. They proposed the adoption of a set of techniques and suitable data structures to reduce the time complexity order, without changing the original GNG semantics. R-tree technique, which is a tree data structures, is used for spatial access methods. The experiment was done on 2, 4, 6 and 8. The result of applying the FGNG and GNG on 8 dimension was not good; so, they tested the proposed method for 2 dimensional data and that what the results of this paper approve it. The aims of this paper is to enhance the time execution of GNG algorithm for 2 dimensional data.

In [20], Fišer, Daniel, et al. suggested two techniques for optimizing GNG; the first one is done to enhance the nearest neighbour search and the second one is done to handle node errors for accelerating the GNG algorithm. However, these two optimization techniques focus only on the efficient execution of the GNG algorithm. They focus on the internal structure of the GNG algorithm instead of modifying the original GNG algorithm. The suggested techniques keep all characteristics of GNG algorithm and make it appropriate to be used on a huge problem.

In [21], García-Rodríguez, José, et al. suggested a fast autonomous growing neural gas (FAGNG) algorithm by modifying the original GNG algorithm. The aim of FAGNG algorithm is to accelerate the GNG algorithm for supporting application of time constraint.

In [22], Guillermo S, et al. suggested to use the evolutionary algorithms (EA) to optimize the parameters of the GNG algorithm when the GNG algorithm deals with 2D image recognition problem.

3. Methodology

3.1. Evolutionary Algorithms

Evolutionary or Genetic Algorithms are a class of optimization algorithms which are based on the principles of Darwinian evolution. Such algorithms are capable of improving/optimizing the solution of a certain problems for which a fitness value can be defined. When several populations are generated which comprised of several individuals (i.e. several possible solutions), the fitness value for each individual (i.e. solution) indicates the suitability of that solution to the subset problem, hence, influences its survival to the next generations. For our implementation of Genetic Algorithm, we used GALIB [23], a C++ based Genetic Algorithm library. GALIB was developed by MIT and it contains a wide range of different types of genomes, configurable genetic operators such as crossover, mutation, selection algorithms, and several terminations conditions.

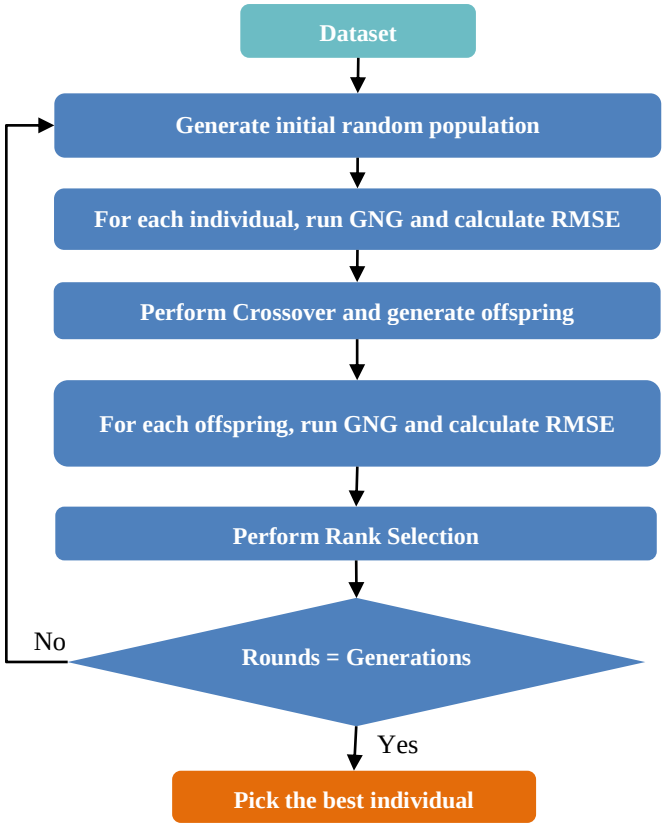

Figure 1. *The flowchart of proposed method PTGNG*

The premise of using the Genetic Algorithm in this study is to tune the parameters of GNG algorithm for making it enable to work properly on multiple dimensions (i.e. 2D, 3D, and 4D). However, GNG algorithm by default uses six main parameters which are:

1. LAMBDA: represents the steps between nodes creation
2. EPSILON_B: defines the adaptation factor for the winning node
3. EPSILON_N: defines the adaptation factor for neighbours of the winning node
4. AGEMAX: sets the maximum age for edges
5. ALPHA: error decrements factor after node insertion
6. BETA: defines the aging factor for nodes

These parameters must be carefully tuned from its default values if an accurate result is required either by applying prior knowledge about the nature of the presented problem to be solved, or by setting them empirically. However, tuning these parameters based on the nature of one particular problem doesn't necessarily ensure the optimum results. Also, the empirical tuning of these parameters values

for each problem can be both extensive and time consuming.

The use of evolutionary optimization algorithm will ensure that these parameters are optimally tuned for any given problem, since it continuously searches and optimizes a vast space of applicable solutions until it reaches to one final good solution. Second, it requires no prior knowledge about the problem itself, therefore it is field or problem independent

3.2 Implementation of the Evolutionary Algorithm for Parameter Optimization

Our proposed PTGNG algorithm is summarized in Fig. 1 such that the evolution cycle starts through randomly generating the values (i.e. genes) for each individual, the random generation function takes two inputs, a minimum value and a maximum value for each individual parameter.

Table 1 describes the minimum and maximum value of each GNG parameter [1,24,25].

Table 1. *The minimum and maximum values of GNG parameters*

Parameter Name	Default value	Min	Max
LAMBDA	30	20	70
EPSILON_B	0.0500	0.0010	1
EPSILON_N	0.0006	0.0001	1
AGEMAX	88	1	1000
ALPHA	0.5000	0	1
BETA	0.0005	0	1.0

Then, for each individual which is an array of real numbers (i.e. chromosome composed of six genes, one gene per-parameter), GNG is tested and the fitness value for that particular chromosome is evaluated and returned.

In this paper Root Mean Square Error (RMSE) is used as a fitness function to evaluate each chromosome/solution. The equation of the RMSE showing below:

$$RMSE = \sqrt{\frac{sum\ of\ squared\ error}{number\ of\ points}} \qquad (1)$$

The returned values are encoded in each chromosome and used later by the Genetic Algorithm, the fitness value represents the base by which individuals are selected/survive onto the next generation. Then, the Genetic Algorithm performs crossover operation and evaluate the offspring using the same procedure described above, and then the best of these offspring are picked using the defined selector for the next round of evolution. Special consideration must be given on how and why the stop condition of the evolutionary algorithm is defined. Two ways can be used as stop conditions:

- By defining a network error threshold (i.e. the minimum fitness value).
- By defining the number of generations the evolutionary algorithm performs.

In our implementation, the second way is used because of three reasons; the first reason is that its simplicity as compare to the first way where network error threshold is hard to be determined, the second reason is that the execution time is lower, the third reason is that, through our experiments on the tested datasets we found that our proposed method doesn't give better solutions after 50 generations.

Fig. 2 – 13 show the results of applying our proposed method to the different datasets described in 2D, 3D and 4D with a different number of generations and populations. Fig. 2 and 3 illustrate the progress of the genetic algorithm in optimizing parameters for the results obtained from applying the Sphere 2D problem. They demonstrate the decreasing RMSE from an initial value of 4.01 at generation 25, reaching stability at 3.99 beyond generations 35 and 45, respectively. Furthermore, Fig. 4 reveals that with a population size of 20, the algorithm rapidly converged to a consistent and optimal solution early on, leading to no significant improvements in subsequent generations. Similarly, Fig. 5-13 consistently demonstrate the genetic algorithm's effective optimization for the results obtained from applying the Tours 2D, Shapes 3D, and Rings 4D problems, with decreasing RMSE values followed by stable performance.

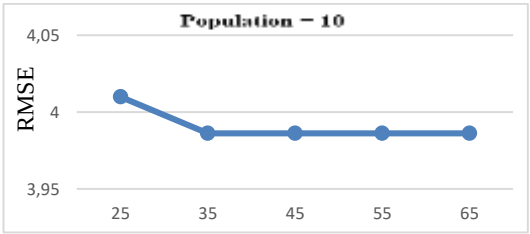

Figure 2. *The results of applying Sphere 2D data with a different number of generations with population=10*

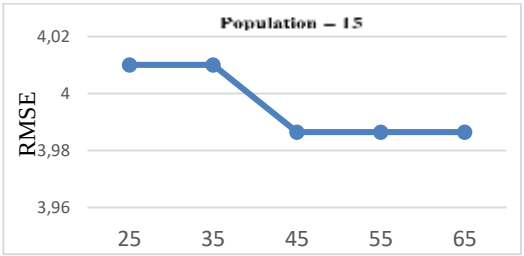

Figure 3. *The results of applying Sphere 2D data with a different number of generations with population=15*

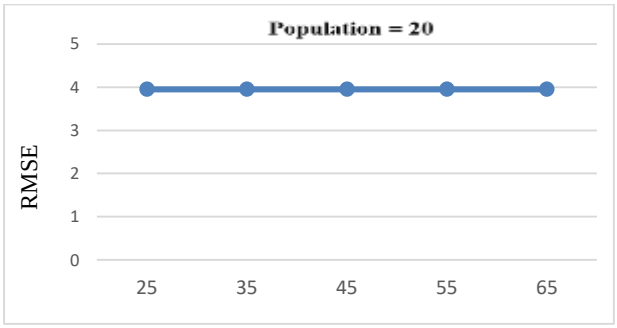

Figure 4. *The results of applying Sphere 2D data with a different number of generations with population= 20*

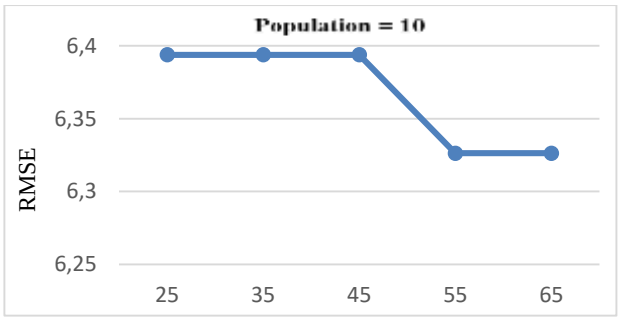

Figure 5. *The results of applying Tours 2D data with a different number of generations with population=10*

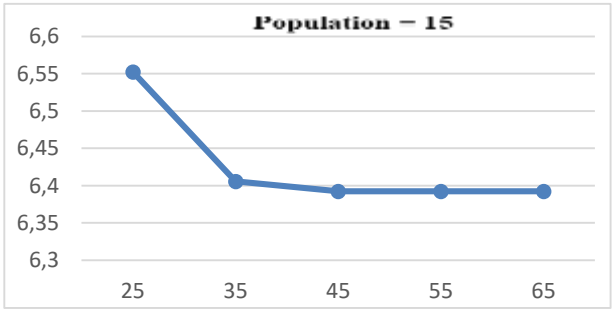

Figure 6. *The results of applying Tours 2D data with a different number of generations with population= 15*

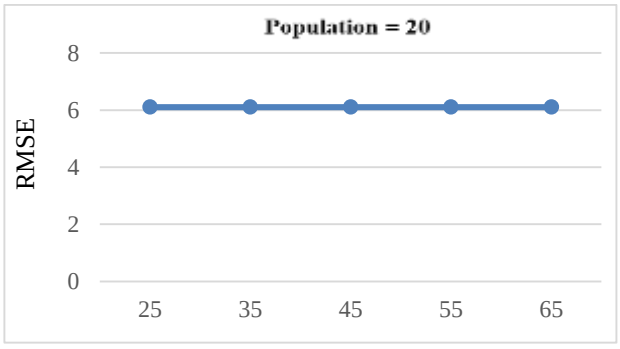

Figure 7. *The results of applying Tours 2D data with a different number of generations with population= 20*

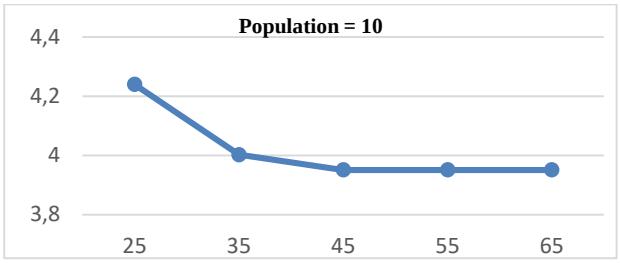

Figure 8. *The results of applying Shapes 3D data with a different number of generations with population= 10*

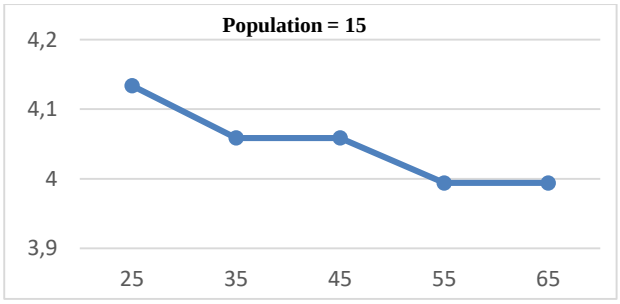

Figure 9. *The results of applying Shapes 3D data with a different number of generations with population= 15*

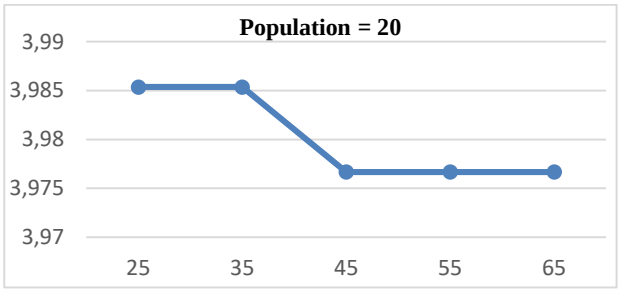

Figure 10. *The results of applying Shapes 3D data with a different number of generations with population= 20*

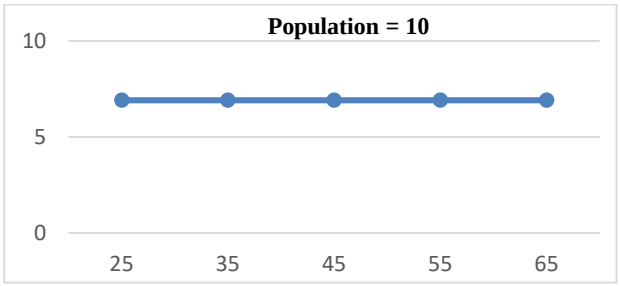

Figure 11. *The results of applying Four rings 4D data with a different number of generations with population= 10*

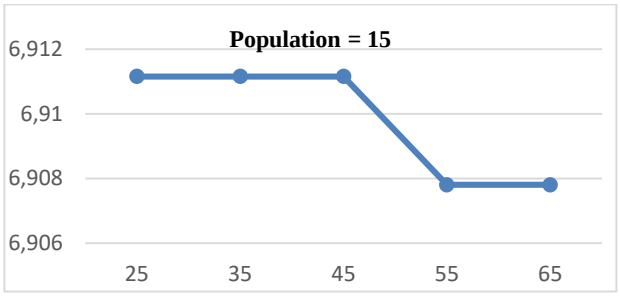

Figure 12. *The results of applying Four rings 4D data with a different number of generations with population= 15*

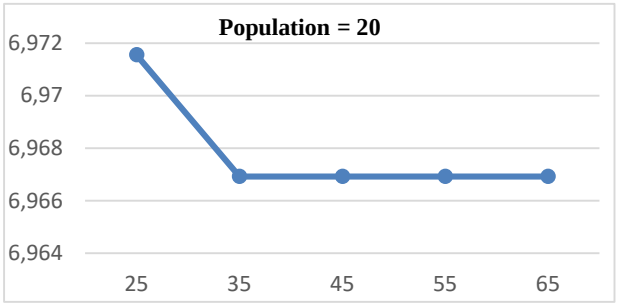

Figure 13. *The results of applying Four rings 4D data with a different number of generations with population= 20*

4. Experimental Study and Discussions

To test the proposed method, we used the following four datasets, four concentric rings, different shapes (rectangle, line, circle and triangle), circle, and a torus, each of which contains 350 points. Each dataset has been drawing by using Excel 2016 as shown in Fig. 14. The other 3D and 4D shapes are also generated randomly using Excel 2016.

From our preliminary test of GNG algorithm on the aforementioned datasets, we found several cases were GNG algorithm goes into an infinite run. In that – for example – when we set the desired number of neurons to be 80 on a dataset of 65 points, GNG algorithm runs infinitely after it reaches a certain number of connected neurons. Such situation is better explained as "when GNG adds new neurons, at the same exact time another neuron in the network is expired", in other words, that specific neuron age passed AGEMAX value. Following on the same dataset, Table 2 presents the desired number of neurons and the correct value of which AGEMAX must be set to. Table 2 shows that for each number of desired neurons, it needs at least AGEMAX value, otherwise the GNG algorithm will not give an appropriate results and it may not stop From the presented values, a linear relationship between the number of desired neurons and AGEMAX value is found and a linear regression line (Fig. 15) has the following equation:

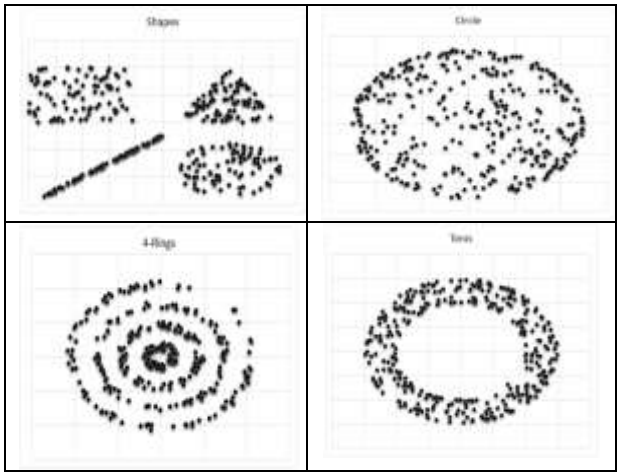

Figure 14. *The four datasets: Shapes, Circle, 4-Rings and Torus*

Table 2. *The desired number of neurons and the correct value of which AGEMAX must be set to.*

Number of Desired Neurons (x)	**Correct *AGEMAX* Value (y)**
30	>=60
40	>=80
50	>=120
60	>=140
70	>=160
80	>=180
90	>=220
100	>=240
110	>=270

$$AGEM = 2.6000\ Number\_of\_desired\_neurons - 18.667 \quad (2)$$

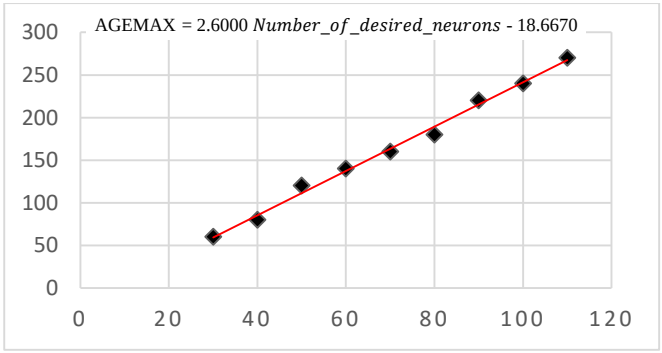

Figure 15. *Liner relationship trend-line between number of desired neurons and AGEMAX values*

With such relation exists between the two variables, we excluded the *AGEMAX* from the evolutionary cycle and the chromosome length changed from seven to six genes only. However, *AGEMAX* is still passed on to GNG algorithm since it is still a requirement, but as a variable derived from the number of desired neurons according to the linear equation. The previous algorithm work with seven

parameters (number of desired neurons, E_b, E_n, AGEMAX. In the proposed method PTGNG by depending on the linear relationship between the AGEMAX and number of desired neurons, we excluded AGEMAX from the set of parameters. Fig. 16 shows the new chromosome after excluding AGEMAX.

# neurons	E_b	E_n	AGEMAX	λ	α	B

(a)

# neurons	E_b	E_n	λ	α	B

(b)

Figure 16. *(a) previous chromosome, (b) new chromosome after applying linear relationship*

For identifying the clustering method with better results, Purity is also used to evaluate the quality of resulting clusters for each of the datasets [26]. It measures how much the cluster has the data of a class (or a cluster in an optimal model). A perfect clustering solution will be the one that leads to clusters that contain data from only a single class (or cluster in an optimal model). The larger the values of Purity, the better the clustering solution is. Purity is defined as:

$$Purity = \frac{1}{n}\sum_{i} max_j(n_{ij}) \qquad (3)$$

Where n is the number of elements in the dataset, n_{ij} represents the number of elements of class j (or a cluster j in an optimal model) in cluster i.

In our experiment, the clusters of an optimal model (or class) can be found by generating several clustering models with different number of clusters using K-means algorithm then the extended contrast index (EC) measure is used to evaluate and select the optimal model [27].

Tables 3, 4, 5, 6, 7 and 8 show the accuracy results, using RMSE and Purity, obtained by applying GNG algorithm to each dataset with number of desired neurons being set to 350, and the proposed method PTGNG along with an optimal number of desired neurons, and the optimized parameters for 2D, 3D and 4D data.

Table 3. *2D datasets:* ***(a)*** *RMSE,* ***(b)*** *Purity,* ***(c)*** *default and optimized parameter values for each dataset.*

Dataset	GNG	PTGNG
Shapes	5.7640	4.0537
Circle	6.2694	3.9415
Four Concerentic Rings	9.0498	7.1466
Tours	6.8910	6.4171

(a)

Dataset	GNG	PTGNG
Shapes	0.9029	0.9686
Circle	0.6057	0.6571
Four Concerentic Rings	0.5971	0.5857
Tours	0.6314	0.6743

(b)

Dataset	Default parameters	PTGNG Optimized Parameters
Shapes	# neurons:350 Eb: 0.05 En: 0.0006 Age: 895 Lambda: 30 Alpha: 0.5 Beta: 0.0005	# neurons:55 Eb: 0.3457 En: 0.5138 Age: 207 Lambda: 21 Alpha: 0.5353 Beta: 0.2355
Circle	# neurons:350 Eb: 0.05 En: 0.0006 Age: 895 Lambda: 30 Alpha: 0.5 Beta: 0.0005	# neurons:185 Eb: 0.8106 En: 0.2485 Age: 463 Lambda: 28 Alpha: 0.0010 Beta: 0.0640
Four concentric Rings	# neurons:350 Eb: 0.05 En: 0.0006 Age: 895 Lambda: 30 Alpha: 0.5 Beta: 0.0005	# neurons: 227 Eb: 0.7099 En: 0.6092 Age: 573 Lambda: 31 Alpha: 0.0010 Beta: 0.4956
Torus	# neurons:350 Eb: 0.05 En: 0.0006 Age: 895 Lambda: 30 Alpha: 0.5 Beta: 0.0005	# neurons: 236 Eb: 0.6987 En: 0.9368 Age: 595 Lambda: 44 Alpha: 0.8468 Beta: 0.2939

(c)

Table 4. *3D datasets:* ***(a)*** *RMSE,* ***(b)*** *Purity,* ***(c)*** *default and optimized parameter values for each dataset.*

Dataset	GNG	PTGNG
Shapes	4.0695	4.0440
Circle	7.0349	6.5702
Four Concerentic Rings	8.6143	7.1652
Tours	7.7065	6.1692

(a)

Dataset	GNG	PTGNG
Shapes3D	0.8657	0.9886
Circle	0.8114	0.8286
Four Concerentic Rings	0.5914	0.6314
Tours	0.5143	0.7200

(b)

Dataset	Default parameters	PTGNG Optimized Parameters
Shapes	# neurons:350 Eb: 0.05 En: 0.0006 Age: 895 Lambda: 30 Alpha: 0.5 Beta: 0.0005	# neurons: 181 Eb: 0.7842 En: 0.6258 Age: 450 Lambda: 58 Alpha:0.1158 Beta: 0.4250

Circle	# neurons:350 Eb: 0.05 En: 0.0006 Age: 895 Lambda: 30 Alpha: 0.5 Beta: 0.0005	# neurons: 77 Eb: 0.4316 En: 0.7367 Age: 182 Lambda: 35 Alpha:0.2123 Beta: 0.6312
Four concentric Rings	# neurons:350 Eb: 0.05 En: 0.0006 Age: 895 Lambda: 30 Alpha: 0.5 Beta: 0.0005	# neurons: 266 Eb: 1 En: 0.0901 Age: 674 Lambda: 40 Alpha: 0.6034 Beta: 0.3895
Torus	# neurons:350 Eb: 0.05 En: 0.0006 Age: 895 Lambda: 30 Alpha: 0.5 Beta: 0.0005	# neurons: 37 Eb: 0.0774 En: 0.5617 Age: 79 Lambda: 43 Alpha: 0.6440 Beta: 0.1357

(c)

Table 5. *4D datasets:* ***(a)*** *RMSE,* ***(b)*** *Purity,* ***(c)*** *default and optimized parameters for each dataset.*

Dataset	**GNG**	**PTGNG**
Shapes	5.8721	4.0477
Circle	7.1254	6.7734
Four Concerentic Rings	8.7718	6.6763
Tours	6.5536	5.7802

(a)

Dataset	**GNG**	**PTGNG**
Shapes4D	0.9114	0.8771
Circle	0.6114	0.7143
Four Concerentic Rings	0.6771	0.7829
Tours	0.5571	0.8800

(b)

Dataset	**Default parameters**	**PTGNG Optimized Parameters**
Shapes	# neurons:350 Eb: 0.05 En: 0.0006 Age: 895 Lambda: 30 Alpha: 0.5 Beta: 0.0005	# neurons: 347 Eb: 0.9439 En: 0.2040 Age: 884 Lambda: 27 Alpha:0.1342 Beta: 0.4871
Hyper sphere	# neurons:350 Eb: 0.05 En: 0.0006 Age: 895 Lambda: 30 Alpha: 0.5 Beta: 0.0005	# neurons: 118 Eb: 1.0000 En: 0.0969 Age: 289 Lambda: 42 Alpha:0.8717 Beta: 0.5854
Four concentric Rings	# neurons:350 Eb: 0.05 En: 0.0006 Age: 895 Lambda: 30 Alpha: 0.5 Beta: 0.0005	# neurons: 52 Eb: 0.9712 En: 0.6639 Age: 119 Lambda: 58 Alpha: 0.4892 Beta: 0.3840
Torus	# neurons:350 Eb: 0.05 En: 0.0006 Age: 895 Lambda: 30 Alpha: 0.5 Beta: 0.0005	# neurons: 16 Eb: 0.3073 En: 0.6819 Age: 25 Lambda: 38 Alpha: 0.2028 Beta: 0.2297

(c)

Table 6. *2D datasets: (a) RMSE with Flip mutation, (b) Purity, (c) default and optimized parameters for each dataset with Flip mutation.*

Dataset	**GNG**	**PTGNG with Flip mutation**
Shapes	5.7640	3.8443
Circle	6.2694	3.9762
Four Concerentic Rings	9.0498	7.1631
Tours	6.8910	6.4171

(a)

Dataset	**GNG**	**PTGNG with Flip mutation**
Shapes	0.9029	0.9057
Circle	0.6057	0.7486
Four Concerentic Rings	0.5971	0.6429
Tours	0.6314	0.6629

(b)

Dataset	**Default parameters**	**PTGNG Optimized Parameters with Flip mutation**
Shapes	# neurons:350 Eb: 0.05 En: 0.0006 Age: 895 Lambda: 30 Alpha: 0.5 Beta: 0.0005	# neurons: 122 Eb: 0.8080 En: 0.7589 Age: 298.3333 Lambda: 40 Alpha:0.9712 Beta: 0.0707
Hyper sphere	# neurons:350 Eb: 0.05 En: 0.0006 Age: 895 Lambda: 30 Alpha: 0.5 Beta: 0.0005	# neurons: 110 Eb: 0.4456 En: 0.3734 Age: 296.3333 Lambda: 53 Alpha:0.9609 Beta: 0.1804
Four concentric Rings	# neurons:350 Eb: 0.05 En: 0.0006 Age: 895 Lambda: 30 Alpha: 0.5 Beta: 0.0005	# neurons: 157 Eb: 0.7257 En: 0.6800 Age: 389.3333 Lambda: 20 Alpha:0.4945 Beta: 0.7722
Torus	# neurons:350 Eb: 0.05 En: 0.0006 Age: 895 Lambda: 30 Alpha: 0.5 Beta: 0.0005	# neurons: 295 Eb: 0.5796 En: 0.3734 Age: 748.3333 Lambda: 21 Alpha: 0.9609 Beta: 0.1804

(c)

Table 7. *3D datasets:* ***(a)*** *RMSE with Flip mutation,* ***(b)*** *Purity,* ***(c)*** *default and optimized parameters for each dataset with Flip mutation.*

Dataset	**GNG**	**PTGNG with Flip mutation**

Shapes	4.0695	3.8967
Circle	7.0349	6.0591
Four Concerentic Rings	8.6143	5.6651
Tours	7.7065	6.3903

(a)

Dataset	**GNG**	**PTGNG with Flip mutation**
Shapes	0.8657	0.9000
Circle	0.8114	0.8714
Four Concerentic Rings	0.5914	0.7314
Tours	0.5143	0.6086

(b)

Dataset	**Default parameters**	**PTGNG Optimized Parameters with Flip mutation**
Shapes	# neurons:350 Eb: 0.05 En: 0.0006 Age: 895 Lambda: 30 Alpha: 0.5 Beta: 0.0005	# neurons: 254 Eb: 0.7355 En: 0.9438 Age: 642.3333 Lambda: 60 Alpha:0.0561 Beta: 0.3773
Hyper sphere	# neurons:350 Eb: 0.05 En: 0.0006 Age: 895 Lambda: 30 Alpha: 0.5 Beta: 0.0005	# neurons: 26 Eb: 0.4284 En: 0.7529 Age: 51.3333 Lambda: 21 Alpha:0.0561 Beta: 0.0707
Four concentric Rings	# neurons:350 Eb: 0.05 En: 0.0006 Age: 895 Lambda: 30 Alpha: 0.5 Beta: 0.0005	# neurons: 26 Eb: 0.5796 En: 0.8279 Age: 51.3333 Lambda: 36 Alpha:0.0742 Beta: 0.0398
Torus	# neurons:350 Eb: 0.05 En: 0.0006 Age: 895 Lambda: 30 Alpha: 0.5 Beta: 0.0005	# neurons: 215 Eb: 0.0979 En: 0.7529 Age: 540.3333 Lambda: 40 Alpha: 0.9712 Beta: 0.7067

(c)

Table 8. *4D datasets:* ***(a)*** *RMSE with Flip mutation,* ***(b)*** *Purity,* ***(c)*** *default and optimized parameters for each dataset with Flip mutation.*

Dataset	**GNG**	**PTGNG with Flip mutation**
Shapes	5.8721	3.9361
Circle	7.1254	6.7735
Four Concerentic Rings	8.7718	6.6763
Tours	6.5536	5.7802

(a)

Dataset	**GNG**	**PTGNG with Flip mutation**
Shapes	0.9114	0.8943
Circle	0.6114	0.8343
Four Concerentic Rings	0.6771	0.7829
Tours	0.5571	0.8400

(b)

Dataset	**Default parameters**	**PTGNG Optimized Parameters with Flip mutation**
Shapes	# neurons:350 Eb: 0.05 En: 0.0006 Age: 895 Lambda: 30 Alpha: 0.5 Beta: 0.0005	# neurons: 260 Eb: 0.2923 En: 0.7529 Age: 657.3333 Lambda: 40 Alpha: 0.3719 Beta: 0.2988
Hyper sphere	# neurons:350 Eb: 0.05 En: 0.0006 Age: 895 Lambda: 30 Alpha: 0.5 Beta: 0.0005	# neurons: 118 Eb: 1.0000 En: 0.0969 Age: 289 Lambda: 42 Alpha: 0.8717 Beta: 0.5854
Four concentric Rings	# neurons:350 Eb: 0.05 En: 0.0006 Age: 895 Lambda: 30 Alpha: 0.5 Beta: 0.0005	# neurons: 52 Eb: 0.9713 En: 0.6639 Age: 119 Lambda: 58 Alpha:0.4892 Beta: 0.3840
Torus	# neurons:350 Eb: 0.05 En: 0.0006 Age: 895 Lambda: 30 Alpha: 0.5 Beta: 0.0005	# neurons: 16 Eb: 0.3073 En: 0.6819 Age: 26 Lambda: 38 Alpha: 0.2028 Beta: 0.2297

(c)

Each chromosome in PTGNG consists of six genes, and each gene's value is represented by a real value. Using real genome in Genetic Algorithms only permits the use of two mutation types, Flip Mutation and Swap Mutation. The first type randomly picks two genes and flip all 0's to 1's, and vice versa. This type of mutation is usually used in the binary application or encoding. The second type, however, swaps the values of the two randomly selected genes. To perform the swap mutation, two alleles are chosen randomly and exchange their locations. Testing Swap Mutation failed to produce any results due to the reason that swapping genes caused the assignments of gene to a value which beyond the predefined Min and Max value for that particular gene (i.e. parameter), hence, GNG failed to build the topological network. However, testing Flip Mutation showed noticeable improvements when compared to previous results. Tables 6, 7 and 8 summarize the obtained results after applying the Flip mutation on the different datasets 2D, 3D and 4D.

5. Results

From the results we obtained from testing the proposed approach, and from comparing the

proposed method to what we found in the literatures, we can conclude the following findings:

- From the approach proposed in [22], we found that the suggested method in this research article covers the optimization of GNG algorithm for multiple dimensions if compared to only two dimensional data. Second, our approach can work on any dataset described in multiple dimensions.
- The execution time for all experiments were between 10s and 2m. However, two of 3D datasets took more than 3m.
- During testing the proposed approach on the aforementioned datasets, we found quite often that there are no better results can be achieved after fifty generations.
- When using two-point crossover approach instead of one point, we only observed few cases of improvements. Also the same occurred when changing the crossover ratio value. Also, using Flip Mutation combined with Single Point crossover showed a noticeable improvement in solution accuracy, but with a huge increase of execution time. We observed a minimum of ~10m and a maximum of ~35m compared to ~15s to 1m.
- From the proposed approach, namely FGNG, presented in 2014 [19], the researchers were interested in optimizing GNG execution time compared to other four published approaches. Our proposed approach in this paper demonstrates relatively fast execution time but still slower when compared to the FGNG. However, reducing the number of generations and populations in the propose method can guarantee a good execution time, but it will be on the cost of producing less optimized results which is beyond the scope of this paper.
- We found a linear relationship between the number of desired neurons and *AGEMAX* parameters. We used such relationship to derive the value of *AGEMAX* from the number of desired neurons, hence, excluding *AGEMAX* from evolution cycle can reduce the occurrence of GNG local optima problem. In our implementation, it was reduced to 10 and sometimes to 16 for every complete evolution cycle.

6. Conclusions

The GNG algorithm is an unsupervised learning algorithm which belongs to the competitive learning algorithms. It was proposed in 1995 by Fritzke as unsupervised learning model and it is a popularization of the GCS algorithm and based on the NG algorithm. What makes such algorithm special from other learning models is that nodes (neurons) are periodically change (adding and removing) according to the need of presented data and to reduce the overall representation error. GNG algorithm has no restrictions if compared to SOM model.

In this paper we proposed and implemented an evolutionary based approach, namely PTGNG, to improve the performance of GNG algorithm by tuning its parameters when it works in multiple dimensional space. Root Mean Square Error (RMSE) is used as a fitness function to evaluate each chromosome/solution. The returned values are encoded in each chromosome and used later by the Genetic Algorithm, the fitness value represents the base by which individuals are selected/survive onto the next generation. At that time, the Genetic Algorithm performs crossover operation and evaluate the offspring using the same procedure described above, and then the best of these offspring are picked using the defined selector for the next round of evolution. Special consideration must be given on how and why the stop condition of the evolutionary algorithm is defined. The idea basically relies on finding the optimum set of parameter values for any given problem to be solved using GNG algorithm. The evolutionary algorithm by its nature searches a vast space of applicable solutions and evaluate each solution individually. The results showed that, after applying our proposed algorithm by tuning the set of GNG parameter values, GNG can represent and capture the structure of data described in multiple dimensional space with a small number of neurons and with a better accuracy.

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

References

[1] Fritzke, B. (1994). A growing neural gas network learns topologies. *Advances in neural information processing systems*, *7*.

[2] Fritzke, B. (1997). Some competitive learning methods. *Artificial Intelligence Institute, Dresden University of Technology*, 100.

[3] Fritzke, B. (1994). Growing cell structures—a self-organizing network for unsupervised and supervised learning. *Neural networks*, *7*(9), 1441-1460. DOI:10.1016/0893-6080(94)90091-4

[4] Martinetz, T. and Schulten, K. (1991), "A" neural-gas" network learns topologies," *Artif. Neural Networks*, pp. 397–402.

[5] Qin, A. K., & Suganthan, P. N. (2004). Robust growing neural gas algorithm with application in cluster analysis. *Neural networks*, *17*(8-9), 1135-1148. DOI:10.1016/s0893-6080(04)00166-2

[6] Andreakis, A., Hoyningen-Huene, N. V., & Beetz, M. (2009). Incremental unsupervised time series analysis using merge growing neural gas. In *Advances in Self-Organizing Maps: 7th International Workshop, WSOM 2009, St. Augustine, FL, USA, June 8-10, 2009. Proceedings 7* (pp. 10-18). Springer Berlin Heidelberg. DOI:10.1007/978-3-642-02397-2_2

[7] Kohonen, T. (1997, June). Exploration of very large databases by self-organizing maps. In *Proceedings of international conference on neural networks (icnn'97)* (Vol. 1, pp. PL1-PL6). IEEE. DOI:10.1109/icnn.1997.611622

[8] Qin, A. K., & Suganthan, P. N. (2005). Enhanced neural gas network for prototype-based clustering. *Pattern recognition*, *38*(8), 1275-1288. DOI:10.1016/j.patcog.2004.12.007

[9] Strickert, M., & Hammer, B. (2005). Merge SOM for temporal data. *Neurocomputing*, *64*, 39-71. DOI:10.1016/j.neucom.2004.11.014

[10] Lomp, O. (2008). Finding Optimal Parameters for Neural Gas Networks Using Evolutionary Algorithms.

[11] Al Shehabi, S., & Lamirel, J. C. (2005, July). Multi-Topographic Neural Network Communication and Generalization for Multi-Viewpoint Analysis. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* (Vol. 3, pp. 1564-1569). DOI:10.1109/ijcnn.2005.1556111

[12] Holland, J. H. (1992). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press. DOI:10.7551/mitpress/1090.003.0007

[13] Nannen, V., & Eiben, A. E. (2007, September). Efficient relevance estimation and value calibration of evolutionary algorithm parameters. In *2007 IEEE congress on evolutionary computation* (pp. 103-110). IEEE. DOI:10.1109/cec.2007.4424460

[14] Maron, O., & Moore, A. W. (1997). The racing algorithm: Model selection for lazy learners. *Artificial Intelligence Review*, *11*, 193-225. DOI:10.1007/978-94-017-2053-3_8

[15] Dobslaw, F. (2010). A parameter tuning framework for metaheuristics based on design of experiments and artificial neural networks. In *International conference on computer mathematics and natural computing*. WASET.

[16] Goldberg, D. E. (1988). Holland, JH. Genetic Algorithms in Search. *Optimization, and Machine Learning. Mach. Learn*, *3*, 95-99. DOI:10.1023/a:1022602019183

[17] TAN, R. K., & Şebnem, B. O. R. A. (2017). Parameter tuning algorithms in modeling and simulation. *International Journal of Engineering Science and Application*, *1*(2), 58-66. DOI:10.1109/cicn.2017.8319375

[18] Ventocilla, E., Martins, R. M., Paulovich, F., & Riveiro, M. (2021). Scaling the growing neural gas for visual cluster analysis. *Big Data Research*, *26*, 100254. DOI:10.1016/j.bdr.2021.100254

[19] Mendes, C. A. T., Gattass, M., & Lopes, H. (2014). FGNG: A fast multi-dimensional growing neural gas implementation. *Neurocomputing*, *128*, 328-340. DOI:10.1016/j.neucom.2013.08.033

[20] Fišer, D., Faigl, J., & Kulich, M. (2013). Growing neural gas efficiently. *Neurocomputing*, *104*, 72-82. DOI:10.1016/j.neucom.2012.10.004

[21] García-Rodríguez, J., Angelopoulou, A., García-Chamizo, J. M., Psarrou, A., Escolano, S. O., & Giménez, V. M. (2012). Autonomous growing neural gas for applications with time constraint: optimal parameter estimation. *Neural Networks*, *32*, 196-208. DOI:10.1016/j.neunet.2012.02.032

[22] Donatti, G. S., Lomp, O., & Würtz, R. P. (2010, July). Evolutionary optimization of growing neural gas parameters for object categorization and recognition. In *The 2010 International Joint Conference on Neural Networks (IJCNN)* (pp. 1-8). IEEE. DOI:10.1109/ijcnn.2010.5596682

[23] Wall, M. (1996). GAlib: A C++ library of genetic algorithm components. *Mechanical Engineering Department, Massachusetts Institute of Technology*, *87*, 54.

[24] Fritzke, B. (1994). Fast learning with incremental RBF networks. *Neural Process. Lett.*, *1*(1), 2-5. DOI:10.1007/bf02312392

[25] Fritzke, B. (1995). Growing grid—a self-organizing network with constant neighborhood range and adaptation strength. *Neural processing letters*, *2*, 9-13. DOI:10.1007/bf02332159

[26] Zaki, M. J., & Meira, W. (2014). *Data mining and analysis: fundamental concepts and algorithms*. Cambridge University Press. DOI:10.1017/cbo9780511810114

[27] Lamirel, J. C., & Al Shehabi, S. (2015). Feature maximization based clustering quality evaluation: a promising approach. In *Trends and Applications in Knowledge Discovery and Data Mining: PAKDD 2015 Workshops: BigPMA, VLSP, QIMIE, DAEBH, Ho Chi Minh City, Vietnam, May 19-21, 2015. Revised Selected Papers* (pp. 210-222). Springer International Publishing. DOI:10.1007/978-3-319-25660-3_18