Combination of Self Organizing Maps and Growing Neural Gas

Lukáš Vojáček[1,2], Pavla Dráždilová[2], and Jiří Dvorský[2]

[1] IT4Innovations, VŠB - Technical University of Ostrava,
17. listopadu 15/2172, 708 33 Ostrava, Czech Republic
[2] Department of Computer Science,
VŠB – Technical University of Ostrava, 17. listopadu 15/2172,
708 33 Ostrava, Czech Republic
{pavla.drazdilova,jiri.dvorsky,lukas.vojacek}@vsb.cz

Abstract. The paper deals with the high dimensional data clustering problem. One possible way to cluster this kind of data is based on Artificial Neural Networks (ANN) such as Growing Neural Gas (GNG) or Self Organizing Maps (SOM). The learning phase of ANN, which is time-consuming especially for large high-dimensional datasets, is the main drawback of this approach to data clustering. Parallel modification, Growing Neural Gas with pre-processing by Self Organizing Maps, and its implementation on the HPC cluster is presented in the paper. Some experimental results are also presented.

Keywords: self organizing maps, growing neural gas, high-dimensional dataset, high performance computing.

1 Introduction

Recently, the issue of high-dimensional data clustering has arisen together with the development of information and communication technologies which support growing opportunities to process large data collections. High-dimensional data collections are commonly available in areas like medicine, biology, information retrieval, web analyse, social network analyse, image processing, financial transaction analysis and many others.

Two main challenges should be solved to process high-dimensional data collections. One of the problems is the fast growth of computational complexity with respect to growing data dimensionality. The second one is specific similarity measurement in a high-dimensional space. Beyer et al. presented in [1] for any point in a high-dimensional space the expected distance, computed by the Euclidean distance to the closest and to the farthest point, shrinks with growing dimensionality. These two reasons reduce the effectiveness of clustering algorithms on the above-mentioned high-dimensional data collections in many actual applications.

The authors propose an effective data clustering algorithm which is based on *Growing Neural Gas* (GNG) [9] and *Self Organizing Maps* (SOM). The computational complexity is resolved by the parallel implementation of GNG with

K. Saeed and V. Snášel (Eds.): CISIM 2014, LNCS 8838, pp. 100–111, 2014.

pre-procesing by SOM. Some technical problems have to be resolved in order to effectively train such kind of neural network using an *High Performance Computing* (HPC) cluster with MPI. In our previous paper [14] we proposed a method of parallel implementation of GNG by using only MPI, but there were limitation to effective parallelization.

2 Artificial Neural Networks

In this section we will describe two type of neural network, the first is Self Organizing Maps and the second is Growing Neural Gas and then we present a combination of SOM and GNG.

2.1 Self Organizing Maps

Self Organizing Maps (SOMs), also known as Kohonen maps, were proposed by Teuvo Kohonen in 1982 [6]. SOM is a kind of artificial neural network that is trained by unsupervised learning. Using SOM, the input space of training samples can be represented in a lower-dimensional (often two-dimensional) space [7], called a *map*. Such a model is efficient in structure visualization due to its feature of topological preservation using a neighbourhood function.

SOM consists of two layers of neurons: an *input layer* that receives and transmits the input information, and an *output layer*, the map that represents the output characteristics. The output layer is commonly organized as a two-dimensional rectangular grid of nodes, where each node corresponds to one neuron. Both layers are feed-forward connected. Each neuron in the input layer is connected to each neuron in the output layer. A real number, or weight, is assigned to each of these connections.

2.2 Growing Neural Gas

The representation of Growing Neural Gas is an undirected graph which need not be connected. Generally, there are no restrictions to the topology. The graph is generated and continuously updated by competitive Hebbian Learning [8,10]. According to the pre-set conditions, new neurons are automatically added and connections between neurons are subject to time and can be removed. GNG can be used for vector quantization by finding the code-vectors in clusters [5], biologically influenced [12], image compression, disease diagnosis.

GNG works by modifying the graph, where the operations are the addition and removal of neurons and edges between neurons.

To understand the functioning of GNG, it is necessary to define the learning algorithm. The algorithm described by Algorithm 1 is based on the original algorithm [4] [5], but it is modified for better continuity in the SOM algorithm.

Remark. The notation used in the paper is briefly listed in Table 1.

Table 1. Notation used in the paper

Symbol	Description
M	Number of input vectors
n	Dimension of input vectors, number of input neurons, dimension of weight vectors in GNG output layer neurons
N	Current number of neurons in GNG output layer
N_{max}	Maximum allowed number of neurons in GNG output layer
n_i	i-th input neuron, $i = 1, 2, \ldots, n$
N_i	i-th output neuron, $i = 1, 2, \ldots, N$
e_{ij}	edge between neurons N_i and N_j for some $i, j = 1, \ldots, N$, where $i \neq j$.
E	set of all edges in GNG
G	undirected graph describing topology of GNG, $G(\{\mathsf{N}_1, \ldots, \mathsf{N}_N\}, \mathsf{E})$
T	Number of epochs
t	Current epoch, $t = 1, 2, \ldots, T$
X	Set of input vectors, $X \subset \mathbb{R}^n$
$\boldsymbol{x}(t)$	Current input vector in epoch t, arbitrarily selected vector from set X $\boldsymbol{x}(t) \in X$, $\boldsymbol{x}(t) = (x_1, x_2, \ldots, x_n)$
$\boldsymbol{w_k}(t)$	Weight vector of neuron N_k, $k = 1, 2, \ldots, N$ $\boldsymbol{w_k}(t) \in \mathbb{R}^n$, $\boldsymbol{w_k}(t) = (w_{k1}, w_{k1}, \ldots, w_{kn})$
N_{c_1}	The first Best Matching Unit (BMU_1), winner of learning competition
N_{c_2}	The second Best Matching Unit (BMU_2), second best matching neuron in learning competition
l_{c_1}	Learning factor of BMU_1
l_{nc_1}	Learning factor of BMU_1 neighbours
e_i	Local error of output neuron N_i, $i = 1, 2, \ldots, N$
B	Number of clusters (C)
C_i	ith cluster witch contains similar input vectors
m_i	Number of input vectors in cluster C_i
$\boldsymbol{Z}_i$	Centroid of cluster C_i
β	Neuron error reduction factor
γ	Interval of input patterns to add a new neuron
$\delta(C_i, C_j)$	Distance metric between C_i and C_j
a_{max}	Maximum edge age
BMU	Best match unit
$d(\boldsymbol{x}, \boldsymbol{y})$	Distance between two vectors
epoch	During training phase are all input data one time used
p, q	Natural numbers

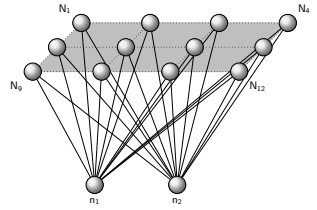
Algorithm 1. Growing Neural Gas algorithm

1. Initialization of network. Two neurons N_1 and N_2 are created, $\mathsf{E} = \{\mathsf{e}_{12}\}$. Weight vectors $\boldsymbol{w_1(t)}$ and $\boldsymbol{w_2(t)}$ are initialized to random values $\boldsymbol{w_{kj}(t)} \in [0, 1] \quad \forall j = 1, \ldots, n, \quad k = 1, 2$.
2. Select arbitrary unused input data vector.
3. Perform the one learning iteration.
4. Reduce error value e_i for all neurons N_i using factor β.
5. Returns to step 2, until all input data vectors have been used.
6. If $t < T$ return to step 2.

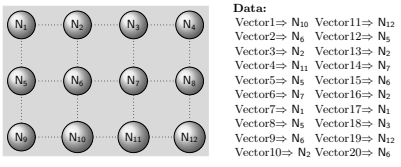
2.3 Combination of SOM and GNG

One of the problem with parallelization which we described in the previous paper [14] is that when adding a new neuron it is necessary to send, in the worst scenario, vectors of two BMU neurons, which takes a lot of time. In this chapter we will describe a method for parallelization of the GNG network, which is based on the distribution of dataset.

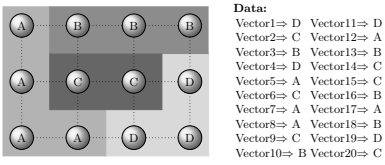
The basic idea is to pre-process the input data by SOM, as a result of which there are clusters of similar data. Subsequently, we created the same number of GNG network as clusters, and assigned each cluster to one GNG. Each GNG creates its own neural map and after the learning process is finished, the results are merged.



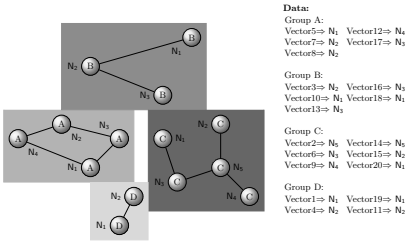
(a) Initialization



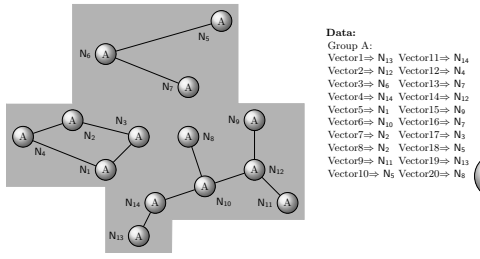
(b) After SOM - Assigned input data



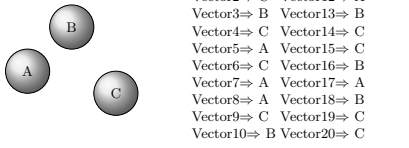
(c) Clusters of similar input data



(d) Separate GNG networks



(e) Completing GNG



(f) Clusters of input vectors

Fig. 1. Combination of SOM and GNG

In general, the whole process is separated into a few phases, where the first phase contains the initialization of the SOM and running of the training process (Fig. 1(a)). In second phase each input vector assigned to one neuron (Fig. 1(b)). The main goal of the third phase is to create clusters of similar neurons (Fig. 1(c)). The clusters are created on the basis of Algorithm 2.

Algorithm 2. Algorithm of creating clusters by create spanning tree

1. Compute the dissimilarity for all $k = 1, \ldots, N$, between each neuron and its neighbors. It is possible to use:
   - Euclidean distance $d_e(\boldsymbol{w}_i, \boldsymbol{w}_j) = (\sum_{k=1}^{n} |\boldsymbol{w}_{ik} - \boldsymbol{w}_{jk}|^2)^{\frac{1}{2}}$,
   - Minkowski distance $d_m(\boldsymbol{w}_i, \boldsymbol{w}_j) = (\sum_{k=1}^{n} |\boldsymbol{w}_{ik} - \boldsymbol{w}_{jk}|^p)^{\frac{1}{p}}$, where $p \in R^+$,
   - cosine dissimilarity $d_c(\boldsymbol{w}_i, \boldsymbol{w}_j) = (1 - \frac{\boldsymbol{w}_i \cdot \boldsymbol{w}_j}{\|\boldsymbol{w}_i\| \|\boldsymbol{w}_j\|})$.
2. Sort dissimilarity from smallest to greatest.
3. Select smallest unused dissimilarity and an associated two neurons.
4. If selected neurons are not in connected component, then create connection between them.
5. If neither of the following conditions is met, return to step 3:
   - The number of the clusters.
   - Number (percentage) of input data in cluster.

This is the end of SOM pre-processing and from now on we will use a GNG network. Now the same number of GNG networks is created as the number of clusters from pre-processing and the clusters are assigned to networks. After running the training process on each GNG network we have learned GNG networks, each of which has a different number of neurons and connections (Fig. 1(d)). All the GNG networks have the same configurations. If the pre-processing is perfect the GNG networks can be completely separated, but to eliminate potential errors we merge all GNG networks into one GNG network, which we do likewise with the input data. After running the training process but without adding a new neuron; only connections between existing neurons are created (Fig. 1(e)). Connected GNG networks create clusters C_i of input vectors.

We implemented following indexes for determining the quality of clusters containing input vectors.

- Davies-Bouldin index [2] – This method is used to determine the distance between clusters and the cohesive. The smaller the resulting value, the better the quality of clusters.

$$S_i = \left(\frac{1}{m_i} \sum_{j=1}^{m_i} |\boldsymbol{x_j} - \boldsymbol{Z}_i|^q \right)^{1/q} \tag{1}$$

$$R_{ij} = \left(\sum_{k=1}^{B} |Z_{ik} - Z_{jk}|^p \right)^{1/p} \tag{2}$$

$$D_{ij} = \frac{S_j + S_i}{R_{ij}} \tag{3}$$

$$DB = \frac{1}{B} \sum_{i=1}^{B} \max_{i \neq j} (D_{ij}) \tag{4}$$

– Dunn index [3] – This method is used to determine the inter-cluster distance (the optimal is maximal) and intra-cluster distance (the optimal is minimal). The range of values are from 0 to 1, where values close to 0 means optimal clustering.

$$\delta(C_i, C_j) = \min_{\boldsymbol{x} \in C_i, \boldsymbol{y} \in C_j} d(\boldsymbol{x}, \boldsymbol{y}) \tag{5}$$

$$\Delta(C_i) = \max_{\boldsymbol{x}, \boldsymbol{y} \in C_i} d(\boldsymbol{x}, \boldsymbol{y}) \tag{6}$$

$$Dunn\ index = \min_{1 \leq i \leq m} \left(\min_{1 \leq j \leq m, j \neq i} \left(\frac{\delta(C_i, C_j)}{\max_{1 \leq k \leq m} \Delta(C_k)} \right) \right) \tag{7}$$

– Silhouette [11] – This method is aimed to evaluation object how well there are assigned to clusters. A range of values are from -1 to 1, where values close to 1 means optimal clustering. For each input vector is calculate silhouette Eq. (11) and then calculated average value Eq. (12).

$$a_i = \sum_{\boldsymbol{x}_i, \boldsymbol{x}_j \in C_k; \boldsymbol{x}_i \neq \boldsymbol{x}_j} \frac{d_*(\boldsymbol{x_i}, \boldsymbol{x_j})}{m_k} \qquad \forall i = 1, \ldots M \tag{8}$$

$$d(\boldsymbol{x}_i, C_j) = \frac{\sum\limits_{\boldsymbol{x}_k \in C_j; \boldsymbol{x}_i \notin C_j} d_*(\boldsymbol{x}_i, \boldsymbol{x}_k)}{m_j} \tag{9}$$

$$b_i = \min_{C_j} d(\boldsymbol{x}_i, C_j) \qquad \forall i = 1, \ldots M \tag{10}$$

$$s_i = \frac{b(i) - a(i)}{\max(a(i), b(i))} \qquad \forall i = 1, \ldots M \tag{11}$$

$$Silhouette = \frac{1}{M} \sum_{i=1}^{M} s_i \tag{12}$$

The entire description above can be summarized as follows: Help speeding up computation parallelization is shown in Fig. 2 where the top layer of parallelization (SOM) is something which the authors described in a previous paper [15].

The results of the SOM algorithm are collected in the process with rank 0, which creates a cluster of input data and assigns the individual processes. Each process (0 to m) creates a custom GNG network that is isolated and only works

with assigned inputs. Subsequently, these neural networks were gathered in the process with rank 0, which performs an epoch over all input data.

Precipitated, obtained parallelization compared to using the GNG algorithm without hierarchy depends on the division of the input data into clusters. In the extreme case where all input data are only in one cluster, the hierarchical solution is less efficient than just the GNG algorithm because it still needs to count the extra time required for the SOM algorithm.

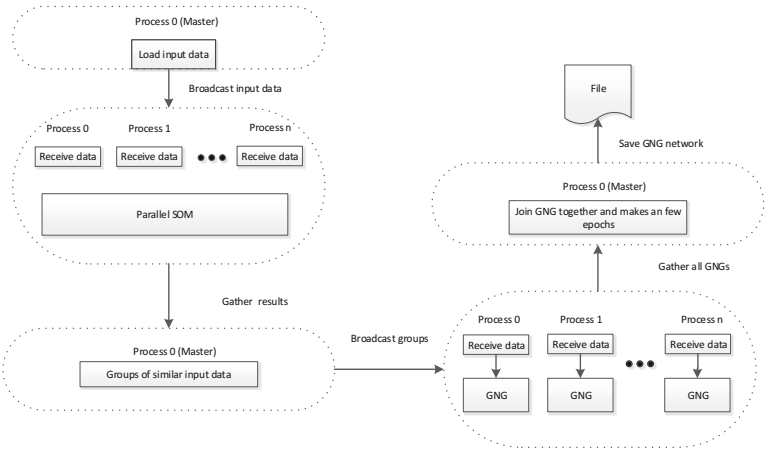


Fig. 2. Parallel Algorithm

3 Experiments

We will describe different datasets and we will provide experiments with datasets in this section.

3.1 Experimental Datasets and Hardware

Two datasets were used in the experiments. The first dataset was commonly used in Information Retrieval – *Medlars*. The second one was the test data for the elementary benchmark for clustering algorithms [13].

Medlars Dataset. The Medlars dataset consisted of 1,033 English abstracts from medical science[1]. The 8,707 distinct terms were extracted from the Medlars dataset. Each term represents a potential dimension in the input vector space. The term's level of significance (weight) in a particular document represents a

[1] The collection can be downloaded from `ftp://ftp.cs.cornell.edu/pub/smart`. The total size of the dataset is approximately 1.03 MB.

value of the component of the input vector. Finally, the input vector space has a dimension of 8,707, and 1,033 input vectors were extracted from the dataset.

Clustering Dataset. Three training data collections called TwoDiamonds, Lsun and Hepta from the Fundamental Clustering Problems Suite (FCPS) were used. A short description of the selected dataset used in our experiments is given in Table 2.

Table 2. Fundamental Clustering Problems Suite – selected datasets

Name	Cases	#Vars	#Clusters	Main Clustering Problem
Target	770	2	6	outlying clusters
Lsun	400	2	3	different variances in clusters
TwoDiamonds	800	2	2	touching clusters
Hepta	212	3	7	clearly separated clusters

Experimental Hardware. All the experiments were performed on a Windows HPC server 2008 with 6 computing nodes, where each node had 8 processors with 12 GB of memory. The processors in the nodes were Intel Xeon 2.27GHz. The topology with the connection between the head node and computing nodes can be found on the web[2] (topology number four). The Enterprise and Private Networks link speed was 1Gbps, the Application link speed was 20Gbps.

3.2 First Part of the Experiment

The first part of the experiments was oriented towards the comparison of algorithms under which is the calculated distance. The used datasets were *Medlars* and *Hepta*. All the experiments were carried out for 20 epochs; the random initial values of neuron weights in the first epoch were always set to the same values. The tests were performed for SOM with rectangular shape 10×10 neurons. The metrics used for determining the quality of clustering is Euclidean distance (in all the algorithms). The achieved quality of clustering is presented in Table 4.

3.3 Second Part of the Experiment

The second part of the experiments was oriented towards a comparison of the standard GNG algorithm and the parallel approach to this GNG learning algorithm with pre-processing by SOM. The *Medlars* dataset was used for the experiment. A parallel version of the learning algorithm was run using 9, 16, 25 and 36 MPI processes. The records with asterisk an (*) represents the results for only one process i.e. this is the original serial learning algorithm and there is no network communication and no SOM algorithm.

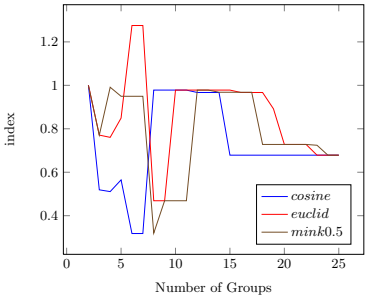
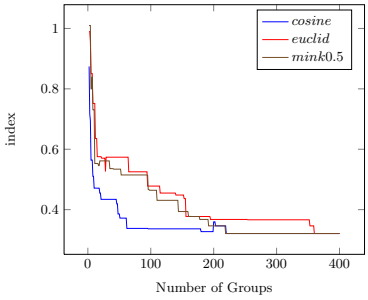
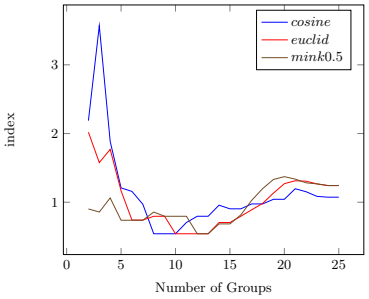
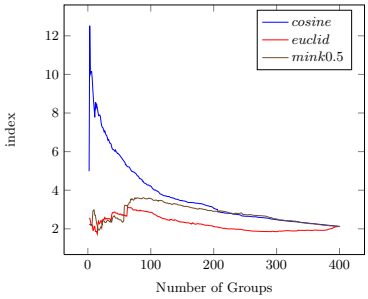
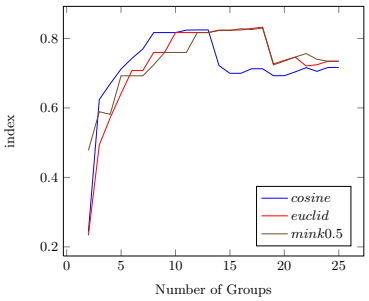
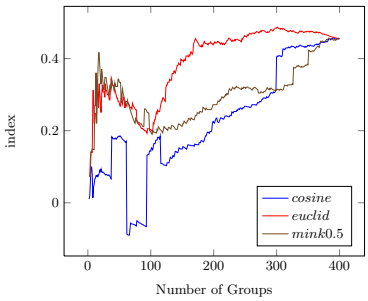
The GNG parameters are the same for all experiments and are as follows $\gamma = 200$, $e_w = 0.05$, $e_n = 0.006$, $\alpha = 0.5$, $\beta = 0.0005$, $a_{max} = 30$, M = 500, $\delta = 200$. The achieved computing time is presented in the Table 3.

[2] `http://technet.microsoft.com/en-us/library/cc719008(v=ws.10).aspx`

Table 3. Computing Time with Respect to Number of Cores, Standard GNG Algorithm, Dataset Medlars

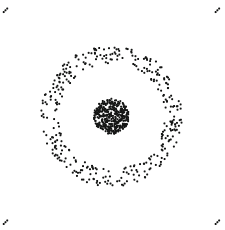
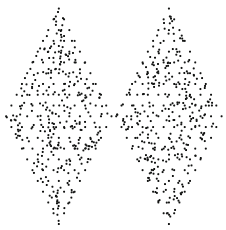
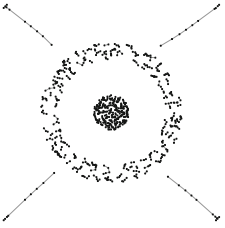
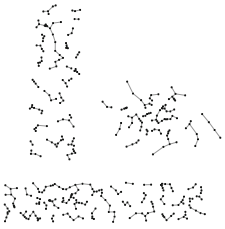
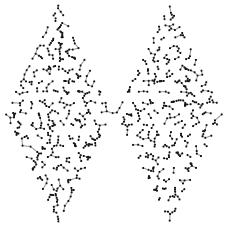
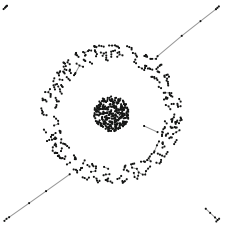
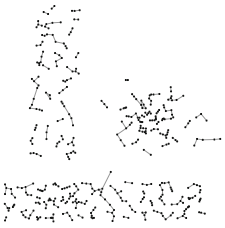
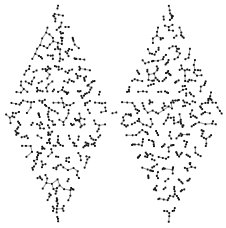
Cores	SOM Dimension	Max # of vectors in one cluster	Computing Time [hh:mm:ss]	
			SOM	GNG
1*	-	1033	-	01:13:41
9	3×3	798	00:00:09	00:45:40
16	4×4	646	00:00:21	00:29:05
25	5×5	229	00:00:25	00:05:55
36	6×6	291	00:00:30	00:08:12

Table 4. Quality of Clustering

	Data Collection	
	Hepta	Medlars
Dunn index		
Davies-Bouldin index		
Silhouette		

As we can see from Table 3, the computing time depends on the number of used core as well. In the latter case the experiment is to see that the time increased even though the reduction was expected. It is caused by the SOM algorithm, which has generated bigger cluster of input vectors for GNG.

Table 5. Graphical representations of data set layout and corresponding GNGs

	Data Collection		
	Target	Lsun	TwoDiamonds
Input data			
Standard GNG			
Parallel GNG using 16 cores			

3.4 Third Part of the Experiment

The third part of the experiments was oriented towards comparing the results obtained in the parallel (with preprocessing by SOM) and standard GNG algorithm. The Clustering dataset was used for the experiment. The parallel version of the learning algorithm was run using 16 MPI processes. The GNG parameters are the same as in the previous experiment.

The first row in the Table 5 shows a layout view of the input data, which are used for training GNG. The outputs of GNG algorithms (without using SOM) are in the second row, which are similar to the parallel version in the last row.

4 Conclusion

In this paper, the parallel implementation of the GNG neural network algorithm is presented. The achieved speed-up was very good and the results from the standard and parallel version of GNG are similar. Thus we can state that operation of the parallel version is optimal. However, the effectiveness of a parallel solution is dependent on the division by SOM. Any improper division may cause a reduction in the quality of the final solution.

In future work we intend to focus on sparse date, and use combinations of neural networks for improved results and improved acceleration.

Acknowledgment. This work was supported by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070) and the national budget of the Czech Republic via the Research and Development for Innovations Operational Programme, and supported by the project New creative teams in priorities of scientific research, reg. no. CZ.1.07/2.3.00/30.0055, supported by Operational Programme Education for Competitiveness and co-financed by the European Social Fund and the state budget of the Czech Republic, and co-financed by SGS, VSB - Technical University of Ostrava, Czech Republic, under the grant No. SP2014/154 'Complex network analysis and prediction of network object behavior.

References

1. Beyer, K., Goldstein, J., Ramakrishnan, R., Shaft, U.: When is nearest neighbor meaningful? In: Beeri, C., Bruneman, P. (eds.) ICDT 1999. LNCS, vol. 1540, pp. 217–235. Springer, Heidelberg (1998)
2. Davies, D.L., Bouldin, D.W.: A cluster separation measure. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-1(2), 224–227 (1979)
3. Dunn, J.C.: A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. Journal of Cybernetics 3(3), 32–57 (1973), http://dx.doi.org/10.1080/01969727308546046
4. Fritzke, B.: A growing neural gas network learns topologies. In: Advances in Neural Information Processing Systems 7, pp. 625–632. MIT Press (1995)

5. Holmström, J.: Growing Neural Gas Experiments with GNG, GNG with Utility and Supervised GNG. Master's thesis, Uppsala University (August 30, 2002)
6. Kohonen, T.: Self-Organization and Associative Memory, 3rd edn. Springer Series in Information Sciences, vol. 8. Springer, Heidelberg (1984, 1989)
7. Kohonen, T.: Self Organizing Maps, 3rd edn. Springer (2001)
8. Martinetz, T.: Competitive hebbian learning rule forms perfectly topology preserving maps. In: Gielen, S., Kappen, B. (eds.) ICANN 1993, pp. 427–434. Springer, London (1993), http://dx.doi.org/10.1007/978-1-4471-2063-6_104
9. Martinetz, T., Schulten, K.: A "neural-gas" network learns topologies. Artificial Neural Networks 1, 397–402 (1991), http://web.cs.swarthmore.edu/~meeden/DevelopmentalRobotics/fritzke95.pdf
10. Prudent, Y., Ennaji, A.: An incremental growing neural gas learns topologies. In: Proceedings. 2005 IEEE International Joint Conference on Neural Networks, IJCNN 2005, July 4-August, vol. 2, pp. 1211–1216 (2005)
11. Rousseeuw, P.J.: Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics 20, 53–65 (1987), http://www.sciencedirect.com/science/article/pii/0377042787901257
12. Sledge, I., Keller, J.: Growing neural gas for temporal clustering. In: 19th International Conference on Pattern Recognition, ICPR 2008, pp. 1–4 (2008)
13. Ultsch, A.: Clustering with SOM: U*C. In: Proc. Workshop on Self-Organizing Maps, Paris, France, pp. 75–82 (2005)
14. Vojáček, L., Dvorský, J.: Growing neural gas – A parallel approach. In: Saeed, K., Chaki, R., Cortesi, A., Wierzchoń, S. (eds.) CISIM 2013. LNCS, vol. 8104, pp. 408–419. Springer, Heidelberg (2013)
15. Vojáček, L., Martinovič, J., Dvorský, J., Slaninová, K., Vondrák, I.: Parallel hybrid SOM learning on high dimensional sparse data. In: Chaki, N., Cortesi, A. (eds.) CISIM 2011. CCIS, vol. 245, pp. 239–246. Springer, Heidelberg (2011)