esf european social fund in the czech republic
EUROPEAN UNION
MINISTRY OF EDUCATION, YOUTH AND SPORTS
OP Education for Competitiveness
INVESTMENTS IN EDUCATION DEVELOPMENT

Growing Neural Gas as Extension of Kohonen Map

Jiří Dvorský

Dept. of Geoinformatics
Palacky Uviversity Olomouc

November 11th 2014

Outline

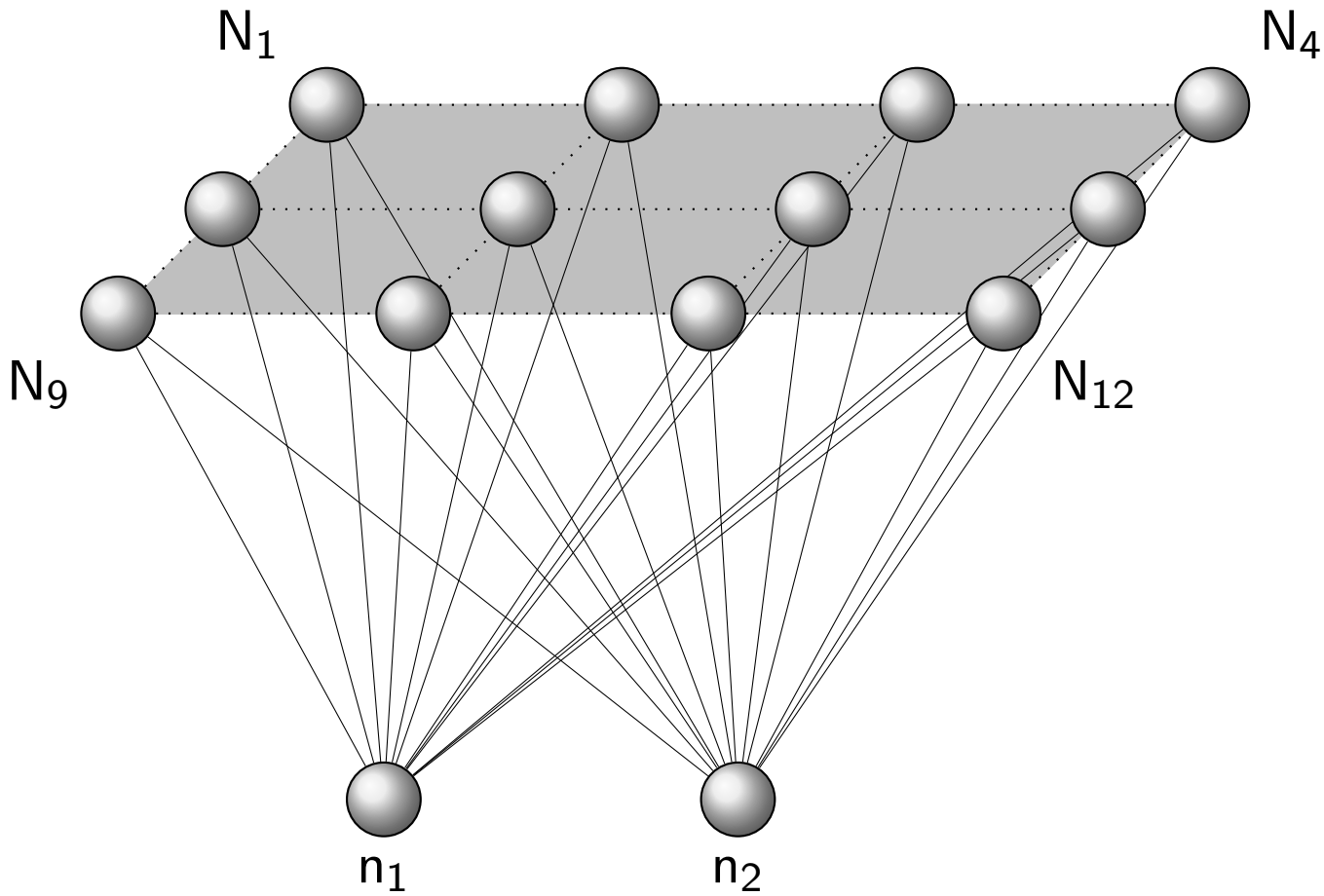
Self Organizing Maps

- also known as Kohonen maps,
- proposed by Teuvo Kohonen in 1982,
- kind of artificial neural network, trained using unsupervised learning,
- the input space of training samples can be represented in a lower-dimensional (often two-dimensional) space, called *map*.
- efficient in structure visualization due to its feature of topological preservation using a neighborhood function.

Self Organizing Maps

- two layers of neurons,
- *input layer* that receives and transmits the input information,
- *output layer*, the map that represents the output characteristics,
- output layer is commonly organized as a two-dimensional rectangular grid of nodes – map.
- 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.

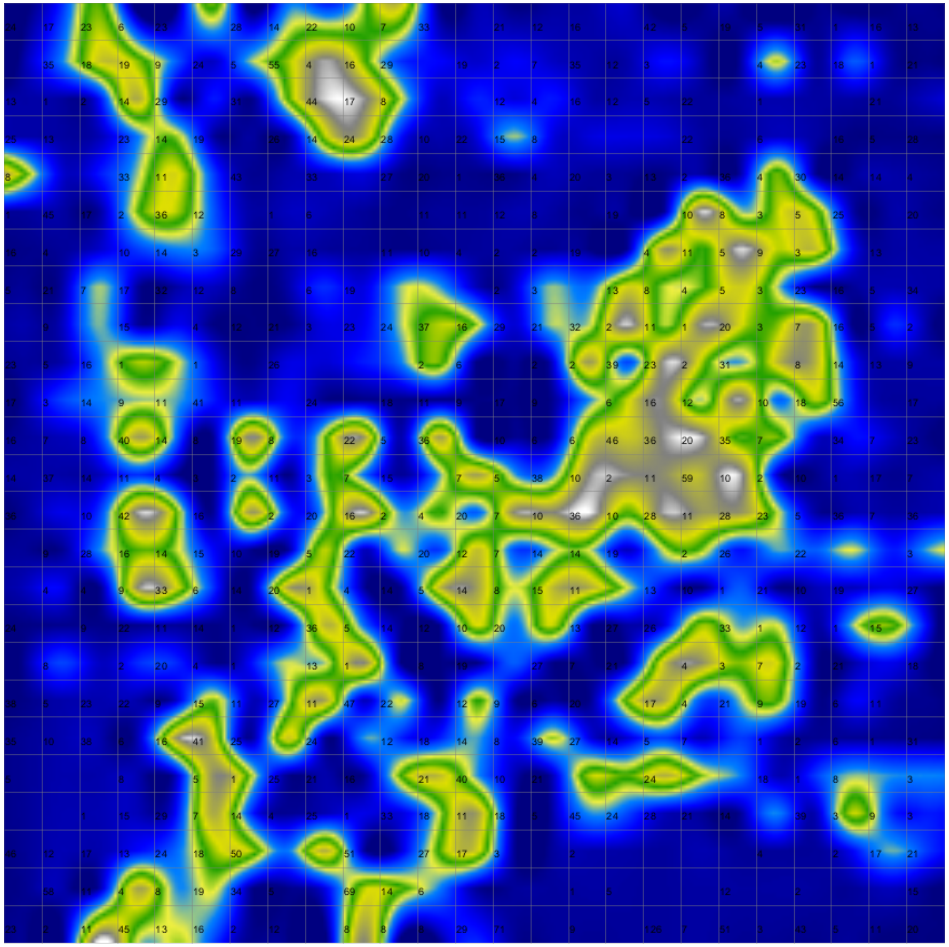
Global view on SOM structure



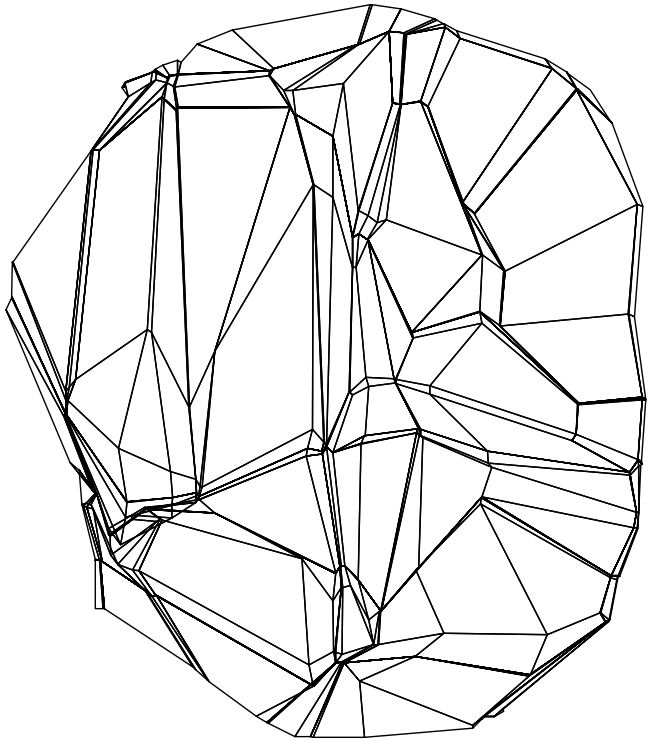
SOM learning

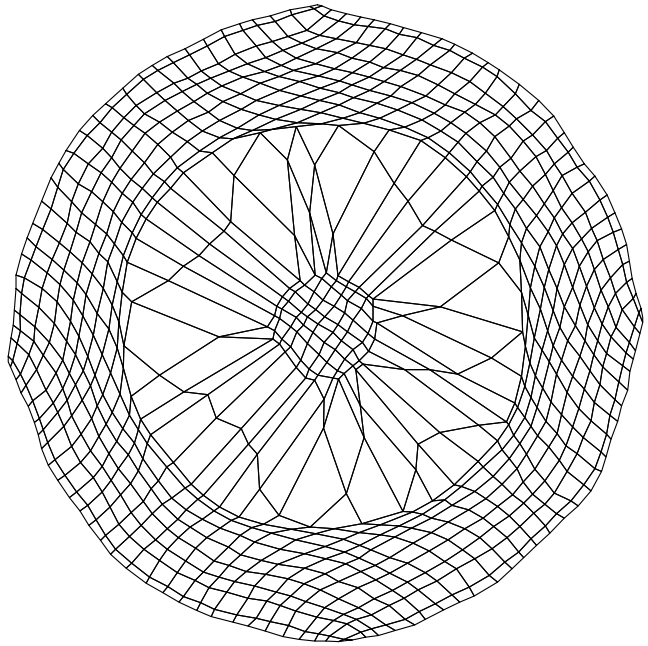
- set of input vectors $\vec{x}$,
- weight of connections (or neurons) $\vec{w}_k$,
- crucial point is finding of the most similar neuron to selected input vectors, *Best Matching Unit* (BMU),
- weight adaptation of neighbours of BMU

SOM – Unified distance matrix



SOM – Connection graph





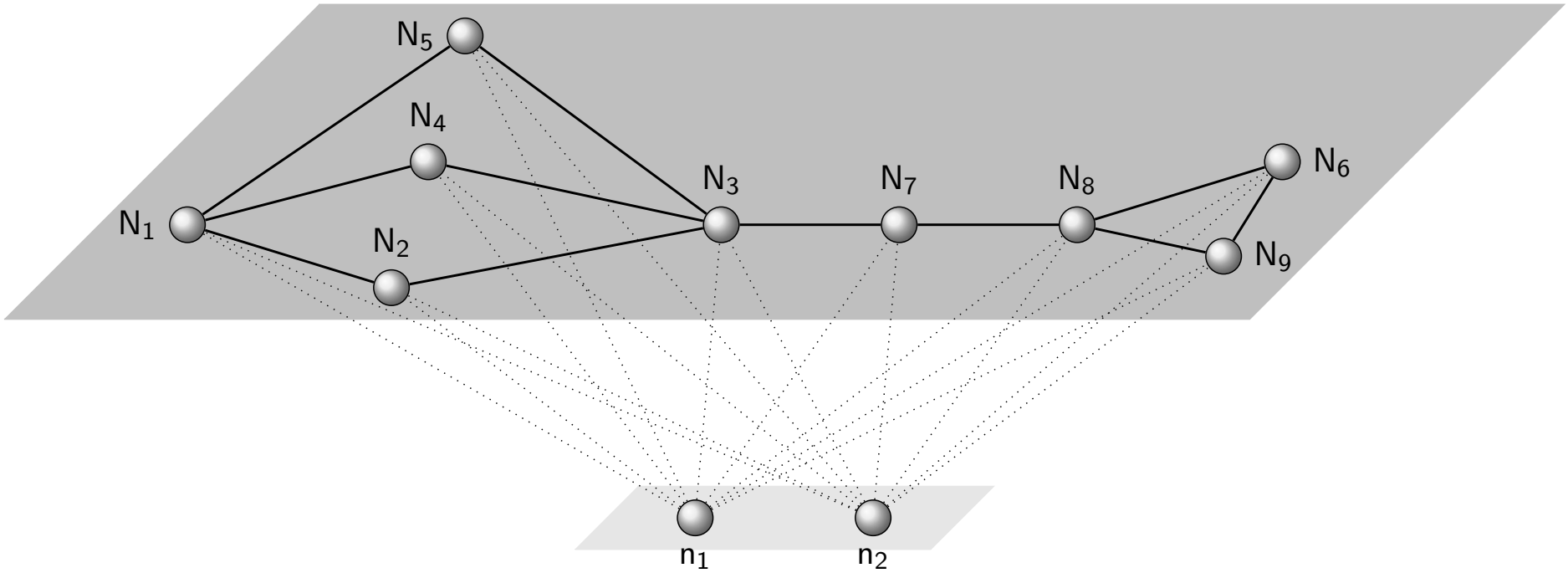
Some drawbacks

- fixed number of neurons, and
- fixed network structure.

Growing Neural Gas (GNG)

- proposed by Fritzke in 1995,
- principle is an undirected graph which need not be continuous,
- generally, there are no restrictions on the topology,
- the graph is generated and continuously updated by competitive Hebbian Learning,
- new neurons are automatically added,
- connections between neurons are subject to time and can be removed.

Example of GNG



Learning of GNG – the overall functionality of GNG

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

Learning of GNG I

1. Find the *Best Matching Unit* (BMU) neurons N_{c_1} and N_{c_2}.
2. Update the local error e_{c_1} of neuron N_{c_1}

$$e_{c_1} = e_{c_1} + \|\vec{w}_{c_1} - \vec{x}\|^2 \tag{1}$$

3. Update the weight vector $\vec{w}_{c_1}$ of neuron N_{c_1}

$$\vec{w}_{c_1} = \vec{w}_{c_1} + \vec{l}_{c_1}(\vec{x} - \vec{w}_{c_1}) \tag{2}$$

4. For all neurons N_k where exists edge $e_{c_1 k}$ (N_{c_1} neighborhood)
   1. Update the weights $\vec{w}_k$ using l_{nc_1} learning factor

$$\vec{w}_k = \vec{w}_k + l_{nc_1}(\vec{x} - \vec{w}_k) \tag{3}$$

   2. Increase age a_{kc_1} of edge $e_{c_1 k}$

$$a_{kc_1} = a_{kc_1} + 1 \tag{4}$$

Learning of GNG II

5. If there is no edge between neurons N_{c_1} and N_{c_2}, then create such edge. If the edge exists, the age is set to 0.
6. If any edge has reached the age of a_{max}, it is removed.
7. If there is a neuron without connection to any edge, the neuron is then removed.
8. If the number of processed input vectors in the current iteration has reached the whole multiple of the value γ and the maximum allowed number of output neurons is not reached, add a new neuron N_{N+1}. The location and error of the new neuron is determined by the following rules:
   1. Found neuron N_b(NBE) which has the biggest error e_b.
   2. Found neuron N_c(NSE) among neighbors of neuron N_b and has the biggest error e_c among these neighbors.
   3. Create a new neuron N_{N+1} and the value of w_n is set as:

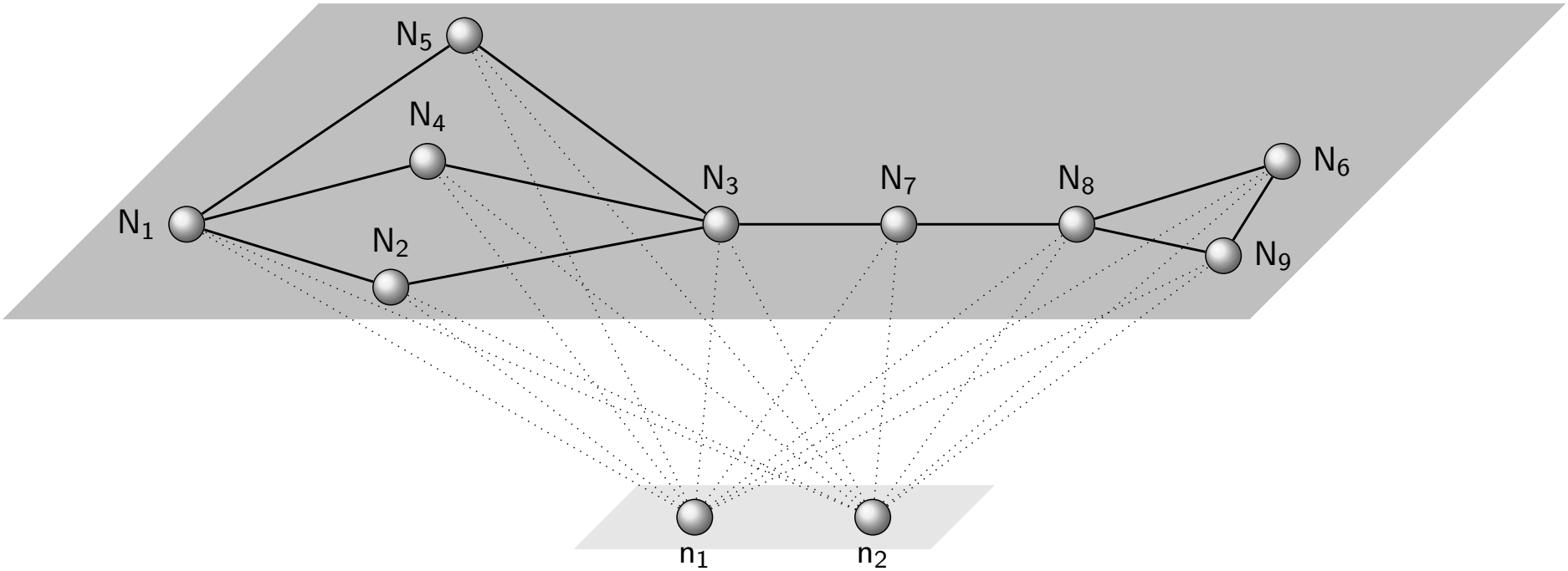
$$\vec{w}_{N+1} = \frac{1}{2}(\vec{w}_b + \vec{w}_c) \tag{5}$$

Learning of GNG III

4. Creating edges between neurons N_b and N_{N+1}, and also between neurons N_c and N_{N+1}.
5. Removed edge between neurons N_b and N_c.
6. Reduction of error value in neurons N_b and N_c using the multiplying factor α. Error for neuron N_{N+1} is equal to the new error of neuron N_b.

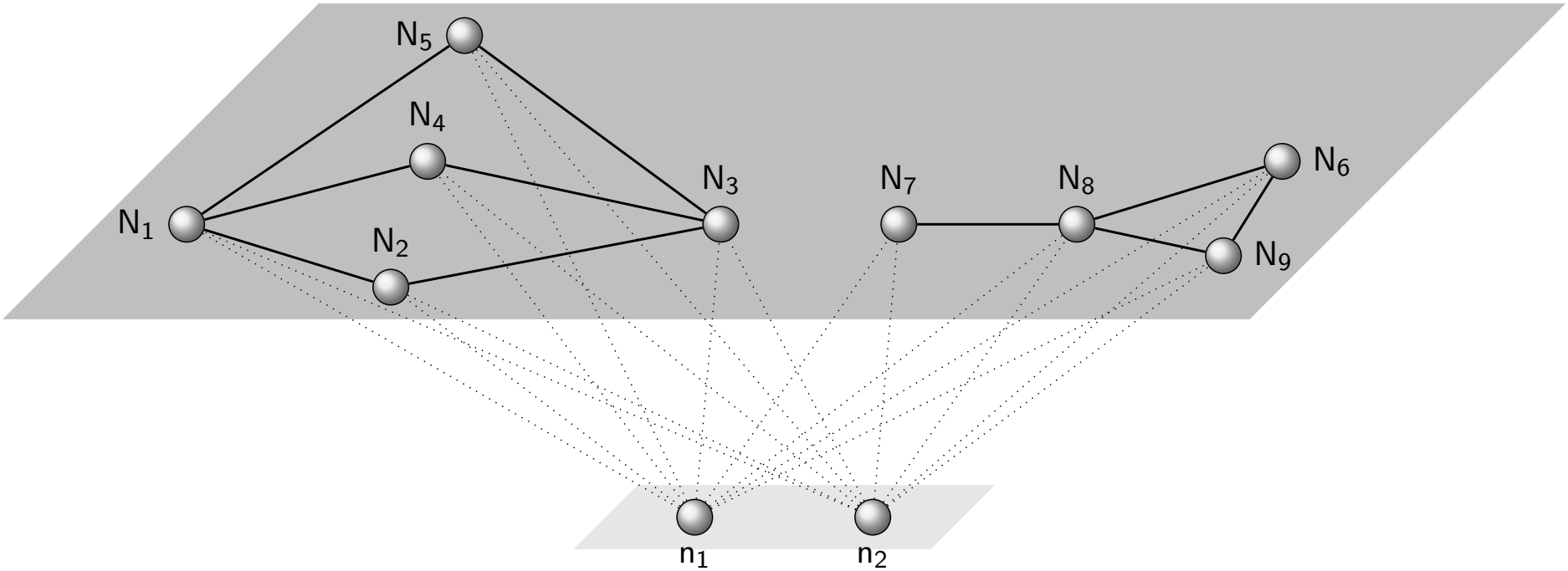
GNG examples

Original condition



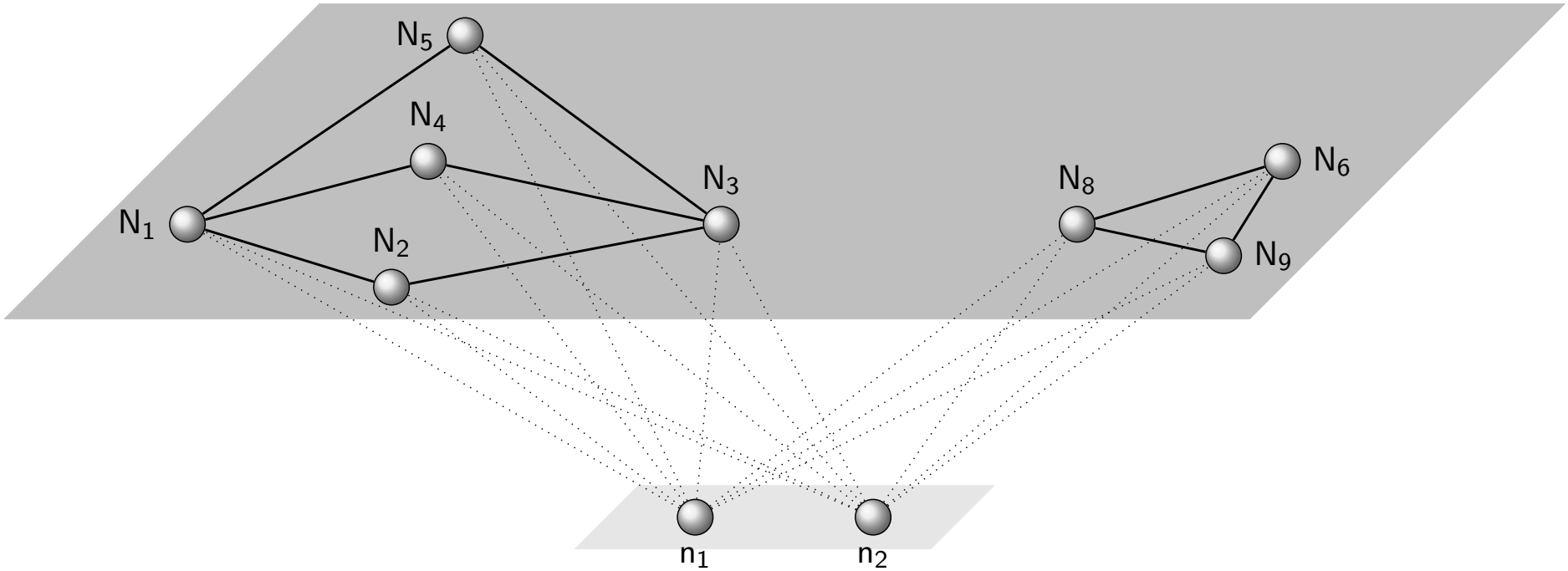
GNG examples

Removal of edges between N_3 a N_7



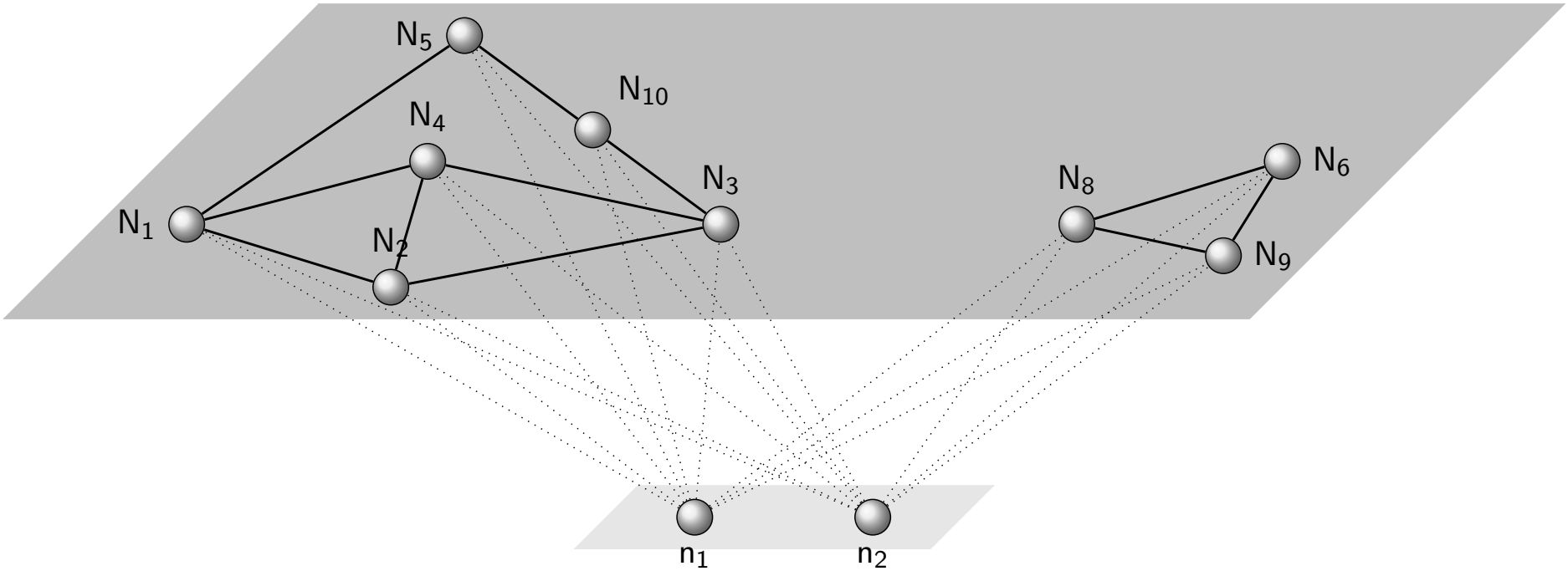
GNG examples

Removal of edges between N_7 a N_8, remove neuron N_7



GNG examples

Added neuron N_{10} between N_5 a N_3, addition of edges between N_2 a N_4



Parallel GNG – how to speed up the learning?

- use multicores computers, computer clusters etc.,
- identify the most time consuming part,
- finding of BMU is then most time consuming,
- the key factor is division of GNG into smaller pieces, parts,
- parts of GNG are assigned to one CPU in ideal case,
- division should keep uniform workload of the CPUs,
- each CPU find its local BMU, then communicate to each other to find global BMU.

Experiments

Fundamental Clustering Problems Suite

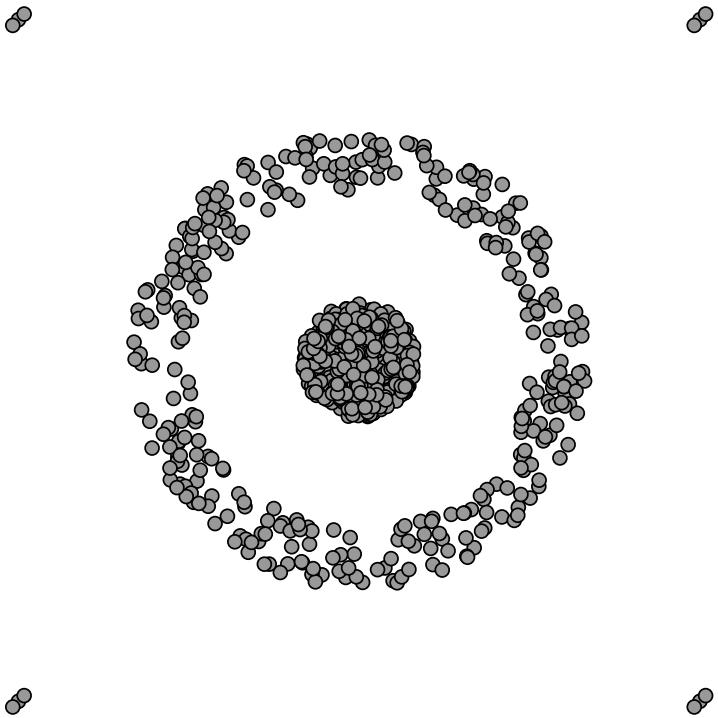
Name	Cases	Number of Variables	Number of Clusters
Target	770	2	6
Lsun	400	2	3

Experimental hardware

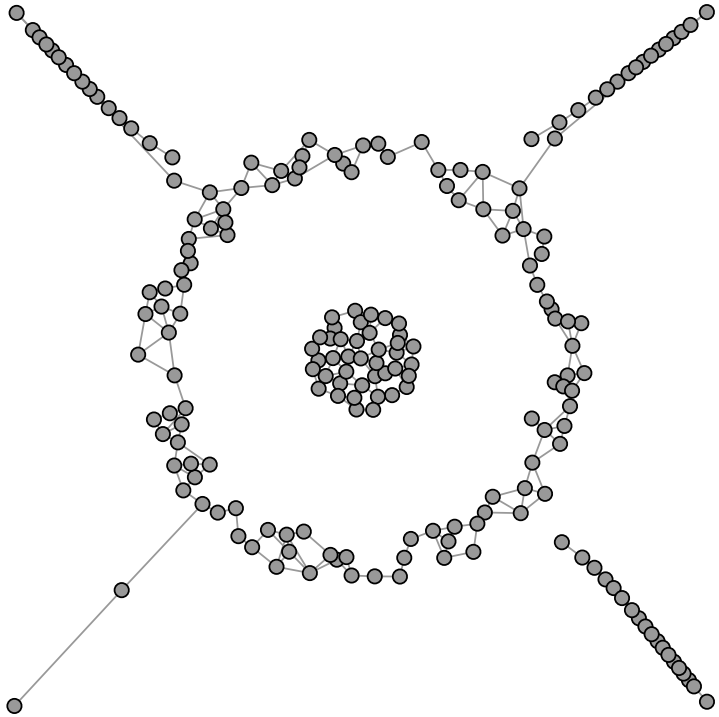
- Windows HPC server 2008 with 2 computing nodes,
- each node had 8 processors with 12 GB of memory,
- total 16 CPUs were used.

Results of dataset *Target*

Input data



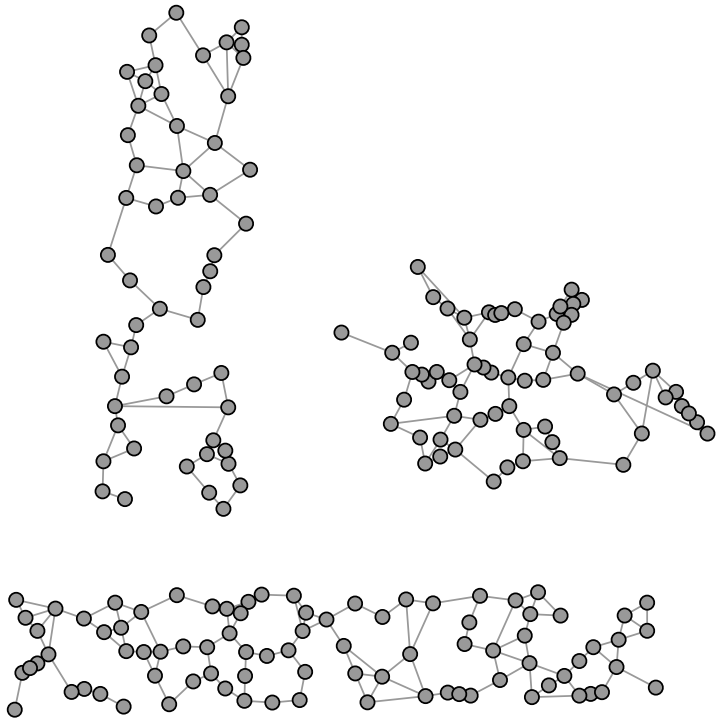
Parallel GNG using 16 cores



Results of dataset *Lsun*

Input data

Parallel GNG using 16 cores



Conclusion

- SOM and GNG
- parallel learning of GNG to speed-up learning process
- parallel GNG produces the same stracture of gas as serial version.

Thank you for your attention