# An High-Order Graph Generating Neural Network

Riccardo Rizzo

**RT-ICAR-PA-03-19** **dicembre 2003**

Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

# An High-Order Graph Generating Neural Network

Riccardo Rizzo[1]

***Rapporto Tecnico N.:***
**RT-ICAR-PA-03-19**

***Data:***
**dicembre 2003**

[1] Istituto di Calcolo e Reti ad Alte Prestazioni, ICAR-CNR, Sezione di Palermo
[2] Università degli Studi di Palermo Dipartimento di Ingegneria Informatica

# An High-Order Graph Generating Neural Network

Rizzo R.
ICAR–Italian National Research Council
viale delle Scienze, 90128 Palermo, Italy
ricrizzo@pa.icar.cnr.it

January 29, 2004

**Abstract**

A large class of neural network models have their units organized in a lattice with fixed topology or generate their topology during the learning process (usually unsupervised). These network models can be used as neighborhood preserving map and some of them generate a perfect topology preserving map of the input manifold using competitive Hebbian rule. But such a structure is difficult to manage if it lays in a high-dimensional space and some hierarchical algorithms were proposed in order to obtain an high-level abstraction of these structures.

In this paper a general structure capable to extract high order information from the graph generated by a large class of self organizing networks is presented. This algorithm will allow to build hierarchical structures starting from the results obtained by using the suitable neural network for the distribution of the input data. Moreover the proposed algorithm is also capable to build a perfect topology preserving map if is trained using a graph that is also a topology preserving map.

## 1 Introduction

A large class of self-organizing network are constituted by a layer of neurons connected between each other in a graph. These networks, from now on GEN networks (from Graph gEnerating Neural networks), are trained in order to adapt their graph to map the input manifold, during the learning stage. The interactions between the neural units are carried by the graph connections that can be created, grow old, and die during the learning stage of the network. These algorithms are inspired by the Self Organizing Network [3] trying to overcome its fixed topology. Application fields of these networks are clustering, adaptive coding of the input values and visualization of high-dimensional data. If the graph structure of the network is fixed the mapping capabilities of these networks are dependent from its topological structure, and the map created by the network will be distorted if this topological structure does not match the one of the input manifold.

A topological preserving map is a representation of the input distribution that preserves the neighboring relationships between the inputs.

In an informal way a structure $G \subset \mathcal{R}^n$ can be defined a topology preserving mapping for a manifold $M \subset \mathcal{R}^n$ if: input vectors of M which are close in $\mathcal{R}^n$ are mapped onto neighboring (or identical nodes in G), and neighboring nodes in G have similar input vectors mapped onto them.

In this paper we will show that topological preserving maps are difficult to manage and visualized when the input data are in an high dimensional space. Depending on the application this can force the user to apply a hierarchical network in which it is possible to choose the level of details to visualize or to use higher levels of the hierarchy to obtain an high level of abstraction.

To avoid this problem a new growing neural algorithm, called HOGEN (High-Order Graph gENerating Network) is proposed. This network can be applied to any GEN network in order to obtain a hierarchical structure. The user can apply the HOGEN network to extract high order information from a trained GEN network and to build a hierarchical structure from a large class of GEN networks.

## 2 GEN Networks and Topology Preserving maps

In this section we will analyze the definition of the topology preserving map and we will see why their characteristics can make difficult to manage the graph created by GEN network.

The concept of topology preserving mapping was introduced by Martinetz and Schulten in [6], we will reprise that discussions and definitions in order to analyze under which conditions a GEN network can create a topology preserving map and which are the consequences of that conditions for visualization and coding purposes. Generally speaking a GEN network can be considered as a graph G with vertices $i = 1, 2, ..., N$ and an adjacency matrix

A with $a_{ij} \in \{0,1\}$; a pointer $\mathbf{w}_i \subset M$, from a set $S = \{\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_N\}$ is connected to each vertex of the graph G. A mapping can be defined by a function $\phi$ from an input manifold $M \in \mathcal{R}^D$ to the graph G, but to obtain a topological preserving map some properties need to be defined.

Let $\phi_s$ be a mapping from M to G defined by:

$$\phi_s : M \longrightarrow G \qquad \mathbf{v} \in M \longrightarrow i^*(\mathbf{v}) \in G$$

with $i^*(\mathbf{v})$ is the vertex for which $v \in V^{(M)}_{i^*(\mathbf{v})}$ is valid, where $V^{(M)}_{i^*(\mathbf{v})}$ is a masked Voronoi polyhedron (the part of the Voronoi polyhedron $V_{i^*(\mathbf{v})}$ that is also part of M).

Let $\phi_s^{-1}$ be the inverse mapping from G to M defined by:

$$\phi_s^{-1} : G \longrightarrow M \qquad i \in G \longrightarrow \mathbf{w}_i \in M$$

The mapping $\phi_s$ is neighborhood preserving if the pointers $\mathbf{w}_i, \mathbf{w}_j$ that are adjacent on M are assigned to vertices *i,j* that are adjacent in G and, vice versa, the mapping $\phi_s^{-1}$ is neighborhood preserving if the vertices *i,j* that are adjacent in G are assigned to locations $\mathbf{w}_i, \mathbf{w}_j$ neighboring on M. Two pointers $\mathbf{w}_i, \mathbf{w}_j$ on M are adjacent if their masked Voronoi polyhedron $V_i^{(M)}, V_j^{(M)}$ share an element $\mathbf{v} \in M$.

The graph G is a topology preserving map if and only if the mapping $\phi_s$ as well as the inverse mapping $\phi_s^{-1}$ is neighborhood preserving.

If we can successfully build a topology preserving map we can guarantee that in the map is preserved the neighborhood relationships between the input points.

In the same paper [6] it is shown that if the distribution S is "dense" on the manifold M, the competitive Hebb rule can form a perfectly topology preserving map of M. The distribution S is "dense" if for each $\mathbf{v} \in M$, called $\mathbf{w}_{i_0} \in S$ the closest unit to $\mathbf{v}$ and $\mathbf{w}_{i_1}$ the second closest, the following condition is satisfied:

$$\triangle\{\mathbf{v}, \mathbf{w}_{i_0}, \mathbf{w}_{i_1}\} \in M \tag{1}$$

i.e. the triangle obtained lies completely on M.

Intuitively this condition is more likely to be true if the number of units $\mathbf{w}_{i_0} \in S$ is "enough big" in order to "cover" completely all the areas of the manifold M. More complex is the shape of the manifold M, more units are necessary to satisfy the condition (1).

Rising the number of nodes in the graph G it will make the visualization and our understanding of the structure of the manifold M more difficult to achieve especially if it lays in a high-dimensional space. Moreover if we consider each element of S as a codeword, and S as a codebook, raising the number of units in S we will obtain a larger codebook where each codeword encodes only very few elements of the input set. This makes each codeword less significant and will diminished the achieved compression. So if it is necessary to approximate the topology of the input manifold with an high degree of precision the map can be only a little simpler than the original manifold and the advantage of using a neural map can be lost.

To avoid this problem usually a hierarchical neural structure is used. These neural networks can be considered as a layered structure so that it is possible to look at higher levels of the layer if a simpler structure is needed and to go in deep if more details are necessary.

## 2.1 Characteristics of the GEN networks

Another growing algorithm is the Growing Cell Structure [1] developed for self–organizing clustering. The GCS algoritm creates a growing k–dimentional simplex distributed over the input manifold (k=1 for a line segment, k=2 for a triangle, k=3 for a tetrahedron and so on). The resulting topology is strictly k-dimensional.

Topology Representing Network (TRN network) [6] has a free structure but a limited number of neural units. It can build a perfect topology representing map if the distribution S is ”dense” on the input manifold. TRN is another of the network that can be used as a input set for the HOGEN. The graph built by the TRN cannot be simplified without loosing the topology representing map feature. Even the number of units should be maintained high in order to satisfy the condition (1).

The constrain on the fixed number of units if removed in another two algorithms: Dynamic TRN and GNG.

The Dynamic TRN was proposed by Sin, Lin and Vuong [7] for learning both topology and clustering information. This model adaptively grows the number of output nodes by applying a vigilance test. A competitive Hebbian rule is applied to learn the global topology information concurrently with the clustering process.

The Growing Neural Gas network (GNG)[4] is another incremental variant of the TRN that has a free topology and no limits on the number of neural units, but its freedom made this structure difficult to manage. To build a simpler graph it is possible to limit the number of neural units, obviously in this way it is difficult to assure the creation of a topology preserving map.

The drawback of this class of algorithm is the same as said before: they can build a topology representing map but the number of neural units that they create makes the neural structure too complex.

A way to obtain a simple graph is to choose a hierarchical GEN structure. These structures are usually constituted by different layers of neurons: each layer is made by an interconnected graph as the GEN network and to each neural unit is associated not only a pointer to the input manifold but also a set of unit of the graph on the lower layer.

A number of hierarchical model were derived from GEN model and proposed in the technical literature, such as SAINT that stems from the SOM and was studied in order to map large set of input patterns, or the HiGCS (Hierarchical GCS) in which each layer is a GCS network.

Another architecture stemmed from GCS is TreeGCS in which the hierarchy is more similar to a tree that is generated during the learning stage using a breath-first search engine.

The HOGEN algorithm is an attempt to propose a general framework for the development of a hierarchical GEN network, the aim of the algorithm is to serve as a general high-level layer of a large category of GEN network, so that the user can choose the GEN algorithm more suitable for the application. The user can adjust the parameter of the network to better approximate the input distribution and start the learning phase. When the learning is finished and the graph of the network is fully developed it is possible to use its graph as input for the HOGEN and obtain an high-order graph that represent an abstraction of the original graph. The two–stage network, the lower layer GEN network and the upper-level HOGEN network works as a hierarchical structure. Fig.1 shows a representation of this structure.

## 3 The HOGEN Algorithm

The basic idea of the proposed algorithm is to extract high-order information (as the shape of the graph or a further clustering of the units) from the complex lattice G generated by the GEN network. This can be accomplished using not only the information contained in the set S of the pointers associated to the vertex of the input graph G, but also considering the information of its adjacency matrix A.

The HOGEN algorithm proceed segmenting the input graph in an adaptive way using a cluster algorithm and after that, linking the obtained clusters in order to build an high-order representation.

As the other growing GEN networks the HOGEN is constituted by a graph $G_H$ that has associated a set of pointers $H = \{\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_{N_l}\}$ to its set of vertex $l = 1, 2, ..., N_l$, and an adjacency matrix $A_H$. The input set for the HOGEN is the graph G with vertices $i = 1, 2, ..., N$, an adjacency matrix A, and the associated pointers $S = \{\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_N\}$ of a generic

GEN network.

## 3.1 Learning algorithm

The HOGEN is a growing network so that the learning phase starts with few neural units, say 2, in a random position inside the manifold M. The learning phase can be divided in cycles, with each cycle composed by an optimization phase, in which the positions of the neural units are optimized to represent the input manifold, and a growing phase, in which new units are added.

During the optimization phase the algorithm optimize the position of the set of pointers H in order to achieve the best representation of the input set S. In this phase all the pointers of the input set are used and the optimization algorithm used is the LBG algorithm that is explained in section 3.2.

After this optimization phase the algorithm checks if it necessary to add new units to the network (growing phase). The conditions checked are related to the portion of the input graph contained inside the Voronoi set $V_{h_l}$ of each unit of the network HOGEN. If $V_{h_p}$ is the Voronoi set corresponding to the pointer $h_p$ of HOGEN network and $V_{h_p}^{(S)}$ is the corresponding masked Voronoi polyhedra, $G_p$ will be the subset of G that contains the vertices in $V_{h_p}^{(S)}$ and all the arches that connect the vertices that have a pointer in $V_{h_p}^{(S)}$. $G_p$ will be called the *associated graph* of the vertex $p$. The gray areas in Fig. 1 highlight the associated graphs of the vertex $p$ and $q$. In the growing phase the graph associated to each HOGEN unit is check for connectivity. This test can be easily made using standard algorithm (i.e. making a breadth–first or a depth–first search of the graph from any vertex and building the corresponding tree; then testing if the tree has the same number of vertex of the graph). In the HOGEN algorithm we can look to a simple binary condition (graph connected or not).

A new unit is added inside in a random position the Voronoi region $V_{h_p}$ if the portion $G_p$ of the graph G inside $V_{h_p}$ is not connected or the number of input units contained in $V_{h_p}$ is more than `max_units`. The new unit will be moved in the optimal position during the optimization cycle. Fig **??** show an explanations of the adding unit method. If a new unit was added to the network then another learning cycle is necessary, if no units are added during the growing phase then the learning is finished and the connection between the units are build.

The links between the units of the HOGEN network are built considering the connections between the sections of the input graph G that are contained inside adjacent Voronoi regions $V_h$. Saying that $h_p$ and $h_q$ are two units of the HOGEN network and $V_{h_p}$ and $V_{h_q}$ the related Voronoi regions, $V_{h_p}$ will contain a subset $G_p$ of the graph G and $G_q$ will be the subset of G contained in $V_{h_q}$. The algorithm will connect $p$ and $q$ if exist at least an arch between a vertex of $G_p$ and a vertex of $G_q$. In fig. 1 the arches that create the link between the units $p$ and $q$ are in bold.

The HOGEN algorithm generates a topology preserving map if the input graph G is a topology preserving map. This is easy to understand if we think that two vertex of the graph G that are linked together can only belong to the same graph $G_p$, or to graph subsets $G_p$ and $G_q$ that are linked together.

Note that the growing phase is performed based on topological considerations and do not involve any error minimization but it is clear that adding a new unit to the HOGEN network modifies the error function and effects the topology of the graph obtained.

The pseudo-code of the learning algorithm is reported in tab.1.

## 3.2 The LBG Algorithm

The LBG algorithm allows the user to build a set of code vectors by moving them to the center of their Voronoi sets. The algorithm converges through a finite number of adaptation steps in a local minimum of the distortion error function. The LBG algortihm will be shown referring the notation to the HOGEN network units, in this case we have a set of $N_l$ units:

$$l = 1, 2, ..., N_l$$

and each of them has a pointer vector $\mathbf{h}_{li} \in M$ where M is the input manifold $M \subset R^n$.

The LBG learning algorithm follows:

1. For each unit $l_i$ of LBG network find its masked Voronoi set

$$V_{li}^{(M)} = \{\mathbf{x} \in M \mid \| \mathbf{x} - \mathbf{h}_{li} \| \leq \| \mathbf{x} - \mathbf{h}_{lj} \| \; l = 1, 2, ..., N_l\}$$

2. Move each unit to the mean of its masked Voronoi set

$$\mathbf{w}_{li} = \frac{1}{\| V_{li}^{(S)} \|} \sum_{\mathbf{x} \in V_{li}^{(S)}} \mathbf{x}$$

3. If during step 3 a unit changes place then go back to step 2, otherwise go to step 5.

4. Return the current set of vector LBG

There are many enhancements to the original LBG algorithm, for example the work proposed in [10]. These enhancements can be used in the learning algorithm of the HOGEN, we just used the standard algorithm to prove the generality of the approach but the algorithm [10] can be helpful to avoid the local minima, or presence of "dead units". However note that in the HOGEN algorithm new units are added to the network during the growing cycle of the learning phase, so the algorithm is less likely to be trapped in a local minima.

### 3.3 Results

The GEN network used to test the HOGEN algorithm is the GNG network. It was chosen because it can grow a perfect topology representing map of the input manifold and its lattice can be very complex especially if the input manifold has a complex shape. The GNG network was trained using some 2-D and 3-D manifolds. To show the topological property of the input manifold and of the output of the networks used (i.e. the shape of its graph in the input space) we superimposed on the same graph the input to the network in gray and the output in solid lines.

The figs. 2.a and 3.a show the 2-D distribution, fig.4.a shows the 3-D distribution that is constituted by random point on the surface of a torus. The GNG network build a map of the input manifold, this map is shown in figs. 2.a , 3.a and fig.4.a in solid lines, the HOGEN is trained using the GNG lattice and the results are shown in figs.2.b, 3.b, and 4.b, where the graph of the GNG network is reported in gray lines because represent the input to the HOGEN network, and the HOGEN graph is reported in solid lines. It is possible to see that a good representation of the input lattice is achieved. Moreover the GNG distribution is segmented to show the portion that belong to the Voronoi region $V_{h_p}^{(S)}$ of each HOGEN unit $h_p$. Note that, comparing fig.2.a and fig.2.b, and fig.3.a and fig.3.b, some links between the units of the GNG networks are missing. The information carried by these links was used to connect the HOGEN units, as explained in paragraph 3.

## 4 Conclusions

Graph extracting neural networks are used in speech and image processing and in robotics where they are useful to map complex input distributions, but they are not useful for visualizing them. This work is an attempt to develop an algorithm that can simplify the graph of these networks by creating clusters of neural units and connecting them in a way that satisfies the original link structure. The HOGEN algorithm creates a superimposed new graph that is a sort of "topological generalization" of the input one and can be used as a tool for visualization. The approach used is simple and effective but needs more investigation; further works on this topic include a real world application to the organization of high-dimensional input manifold (e.g. information space organization [5]).

## References

[1] Fritzke B. "Growing Cell Structures-a self-organizing network for unsupervised and supervised learning", Neural Networks , 7, (9), 1441-1460.

[2] Linde Y., Buzo A., Gray R. M., "An Algorithm for Vector Quantizer Design", IEEE Transactions on Communication, COM-28:84-95, 1980.

[3] Kohonen T., "Self Organizing Maps", Springer Verlag

[4] Fritzke B., "A growing neural gas network learns topologies", NIPS 1994, Denver.

[5] Rizzo R. "Self Organizing Networks to Map Information Space in Hypertext Development", Proceedings of the International ICSC/IFAC Symposium on Neural Computation NC'98, September 23-25, 1998, Vienna, Austria.

[6] Martinetz T., Schulten K., "Topology Representing Networks", Neural Networks, vol.7, n. 3, pp. 507-522, 1994

[7] Si J., Lin S., Vuong M.A., "Dynamic Topology Representing Networks", Neural Networks 13 (2000), pp. 617-627.

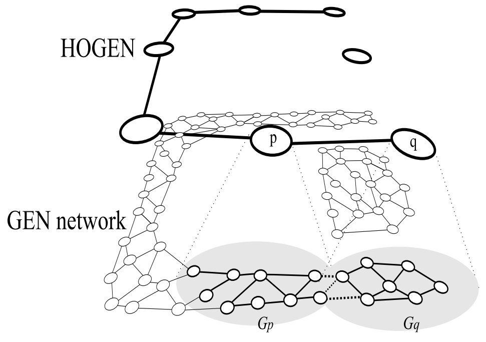
[8] Blackmore J., Miikkulainen, (2000) "Visualizing high–dimensional Structure with Incremental Grid Growing Neural Network" in A. Predits & Russel, "Machine Learning: Proceedings of the 12th International Conference", pp 55-63.

[9] Alahakoon D., Halgamuge S. K., Srinivasan B., "Dynamic Self-Organizing Maps with Controlled Growth for Knowledge Discovery", IEEE Transaction on Neural Networks, vol. 11, n.3, may 2000.

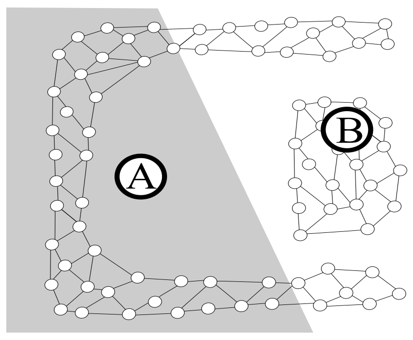
[10] Patane' G., Russo M., "The Enhanced LBG Algorithm", Neural Networks, 14 (2001), pp. 1219–1237.

**Table 1**:The Proposed HOGEN algorithm

```
1 Initialize an LBG network with two units $l_1$ and $l_2$
2 While end = = false
    2.1 Update the position of the HOGEN units according to the LBG
        algorithm, and use the pointers set attached to the input graph G
        $\mathbf{w}_i \in S$ as input ;
    2.2 For each unit $l_i$ of the HOGEN network
        2.2.1 Consider the associated graph $G_i$ i.e. the subset of G made
              by the vertices that have a pointer in $V_{h_i}^{(S)}$ and an adjacency
              matrix $A_{li}$
              $$A_{li} = \{(i,j) \in A \mid i,j \in V_{li}^{(S)}\}$$
        2.2.2 If the graph $G_i$ is not connected or $\parallel V_{li}^{(S)} \parallel >$ max-units then
              add a new unit to the HOGGN network inside the Voronoi
              masked region $V_{li}^{(S)}$ .
    2.3 If no neuron was added then end = true
3 Update the position of HOGEN units according to the LBG algorithm.
4 Create the link between the HOGGN neural units
    4.1 Scan all the couples $l_i, l_j$ of units of HOGEN network and their
        associated graph $G_i$ and $G_j$ and connect $l_i$ and $l_j$ if, in the graph
        G exist an arch that connects a vertex of $G_i$ to a vertex $G_j$.
```

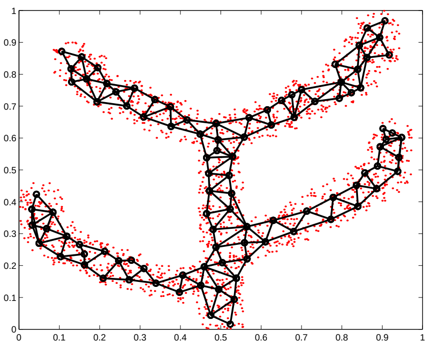


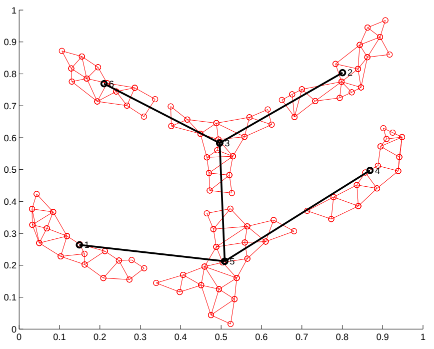
a)



b)

Figure 1: a) HOGEN layer and GEN layer. The gray areas highlight the associated graph of the units p and g. b) A new unit will be added inside the Voronoi region of the unit B ecause the associated graph is not connetted.


a)



b)

Figure 2: a) The "cactus" input distribution and the GNG approximation. b) The HOGEN high–order structure extraction and the associated graph to each HOGEN unit `max--unit=25`

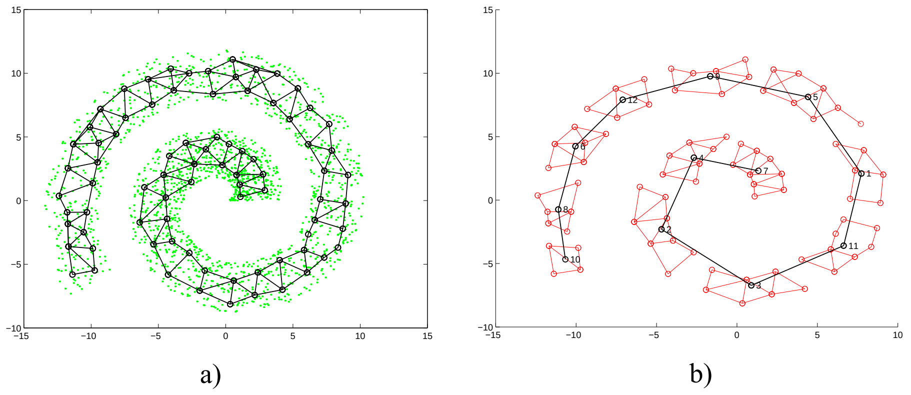


Figure 3: a) The ”spiral” input distribution and the GNG approximation. b) The HOGEN high-order structure extraction and the associated graph to each HOGEN unit `max--units=10`

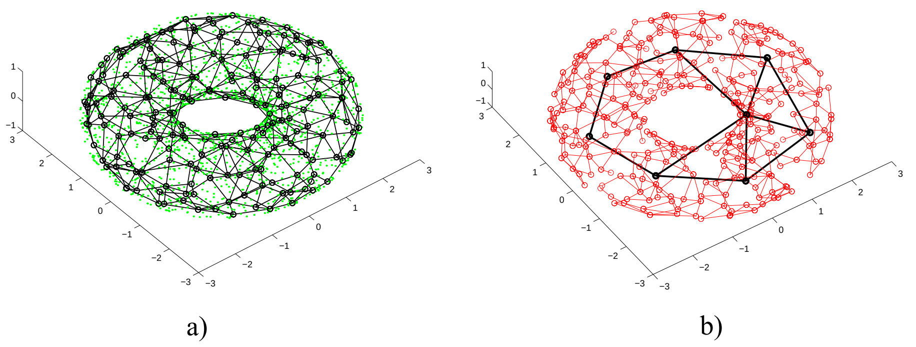


Figure 4: a) The GNG approximation of a 3-D torus structure. b) The HOGEN high–order structure extraction from the torus GNG structure.