Visualizing Multivariate Network on the Surface of a Sphere

Yingxin Wu[1] **Masahiro Takatsuka**[2]

School of Informatics Technologies
The University of Sydney,
NSW 2006 Australia
Email: [1]`chwu@it.usyd.edu.au` [2]`masa@vislab.net`

Abstract

A multivariate network is a graph whose nodes contain multi-dimensional attributes. We propose a method to visualize such a network using spherical Self-Organizing Map (SOM) and circular layout. The spherical SOM produced an initial graph layout by grouping nodes with similar attributes to adjacent areas on the sphere. The circular layout algorithm fine tunes the position of each node to avoid node-node/node-edge overlap and minimize the number of edge crossings. We also implement an interface to project the spherical image onto a 2D plane such that the user can have a global view of the data set. The user is allowed to select any point of interest to be the center of the 2D projection.

Keywords: Multivariate Network, GeoSOM, Graph Drawing, Circular Layout, Information Visualization

1 Introduction

Two distinctive types of data are of particular interest to the information visualization community. One is multi-dimensional data and the other is relational data. Techniques for visualizing these two types of data are very different. For multi-dimensional data, techniques focus mainly on constructing low-dimensional representations (typically 2D or 3D) that preserve certain property of the original data — either preserving distance or topology. The most popular methods include Principle Component Analysis(Hotelling 1933), Sammon's mapping(Sammon 1969) and Self-Organizing Maps(Kohonen 2001). For relational data, researchers emphasize the readability of the layouts — the position of the nodes or bending of the edges are optimized according to certain aesthetic criteria. These criteria reflect the preference of human perception, such as symmetries and few edge crossings. Force directed approaches such as spring embedder(Eades 1984), Kamada and Kawai's algorithm(Kamada & Kawai 1989) are the most common approaches to layout general graphs.

Advances have been made in each of these two fields. However, algorithms rarely take both data types into consideration. In the real world, quite a lot of data contain both relational information and multivariate attributes. For example, in a world trade network, countries are represented as nodes and their trading as edges. Furthermore, each country has attributes like gross domestic product (GDP), GDP growth, population and population growth. Such networks are generally called multivariate networks. Using existing visualization method, it is difficult to reveal the two aspects of the networks at the same time. Hence, answers to either of these questions won't be straightforward:

- Do major trading partners have similar attributes?
- Are countries with similar attributes major trading partners?

To address this deficiency, separate windows have been used to display these information. For example, the GGobi system draws a telecommunication network in one window while the attributes of each node are display in another window (Swayne, Buja & Lang 2003). Selecting some data in one view causes the corresponding data in the other view to change color. The disadvantage of this method is that the user needs to compare two pictures to comprehend the data. We are introducing a method to visualize all the information in one picture. The basic idea is first to use SOM to find an initial layout of the graph. We alter the training process of SOM to consider both the multi-dimensional attributes and graph relations. We then use a graph drawing algorithm to produce a better layout.

The reminder of this paper is organized as follows. Section 2 briefly introduces the self-organizing map and its advantages over other dimensional reduction methods. Section 3 discusses why we use *spherical* self-organizing map and layout the graph on the sphere. Section 4 presents the details of our method and section 5 describes the mechanism for projecting spherical images onto a 2D plane using a cartographic approach. Conclusions and future work are presented in section 6.

2 Self-Organizing Maps

Self-Organizing Map(Kohonen 2001) is an widely used artificial neural network. The neurons are usually organized as 2D rectangular/hexagonal grid. Each neuron represent a set of inputs with a weight vector $\vec{w}_i$. Before training, the components of each weight vector are initialized as random numbers. Training of a SOM is based on competitive unsupervised learning: when an input $\vec{x}$ arrives, the neuron c whose weight vector is nearest to the input win the competition:

$$c = \arg\min_i \|\vec{x} - \vec{w}_i\|^2 \quad (1)$$

Then the winning neuron together with its neighbors update their weight vectors to represent the data better. The update amount is changing with time t:

$$\vec{w}_i(t+1) := \vec{w}_i(t) + \alpha(t) * h_{ci}(t) * (\vec{x}(t) - \vec{w}_i(t)), \quad (2)$$

Here $\alpha(t)$ is the learning rate which decreases with time; $h_{ci}(t)$ is a decreasing function of the distance between the winning neuron c and its neighbors i on the grid. All the neighbors within a radius r of the winning neuron are updated. Usually, the update radius is large at the beginning of the training and shrinking with time. Due to the training process, SOM behaves like a "elastic net" unfolding into the high-dimensional data space. Neighboring neurons tend to model similar regions of the data space, which is SOM's topology preserving characteristic.

In SOM, point density $p(m)$ means how many neurons SOM uses to represent a certain area in the high-dimensional space. Ritter(Ritter 1991) proved that the asymptotic point density for the one-dimensional SOM with one-dimensional input is:

$$p(m) \propto p(x)^{\frac{2}{3}-\frac{1}{3\sigma^2+3(\sigma+1)^2}} \tag{3}$$

where σ is the update radius. This result shows that the SOM roughly follows the density of the training data: the higher the density, the bigger number of neurons are allocated to represent that region in high-dimensional space. Because each neuron occupies the same amount of area on the 2D grid, the total area allocated to represent a cluster on the SOM is somehow proportional to the density of the cluster.

This characteristic makes SOM more suitable for our task than other linear/nonlinear projection methods: for graph drawing, dense areas require more resolution to make the graph readable. Approaches like Principle Component Analysis(PCA) try to maintain the pairwise distance between the data as faithfully as possible. So the area allocated to a cluster on the 2D data map is proportional to the diameter of the cluster in high-dimensional space. This phenomenon is shown in Figure 1. It shows the 2D projections of the same data set using PCA and SOM (Vesanto 2002). Different colors represent different data clusters. Although the yellow and orange cluster have the same amount of data as the pink cluster, they are projected to a very tiny area on the data map produced by PCA, while the area of these three cluster are approximately the same on SOM.

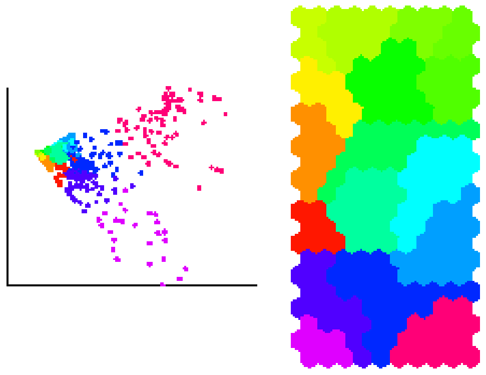


Figure 1: Left: Projection using Principle Component Analysis; Right: Projection using Self-Organizing Map.

3 Spherical Self-Organizing Maps and Spherical Graph Drawing

3.1 Spherical SOM Has Lower Quantization Error

The quality of a SOM can be measured by quantization error, which is the average distance between an input x_i and its best matching neuron w_{bi}:

$$E_q = \frac{1}{N} \times \sum_{i=1}^{N} \|\vec{x}_i - \vec{w}_{bi}\| \tag{4}$$

where N is the number of inputs. Generally speaking, lower quantization error indicates better quality.

The 2D SOM has a notorious "border effect": neurons at the boundaries of the 2D grid have fewer neighbors than those inside the map, so they have less chance to be updated. This not only lowers the SOM's accuracy, but also makes it harder to converge. One solution is to implement the SOM on a spherical grid, thus removing all the boundaries. In (Wu & Takatsuka 2005) we developed the GeoSOM, which used icosahedron-based geodesic dome as the lattice (see Figure 2). Several experiments were carried out to compared the GeoSOM and the 2D SOM. The GeoSOM can reduce about 25% of the quantization error while running at approximately the same speed as the 2D SOM.

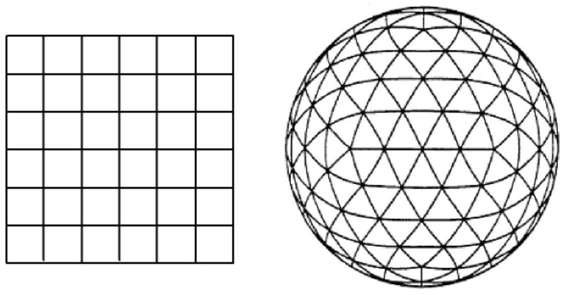


Figure 2: A rectangular grid and a four frequency icosahedron.

3.2 Spherical SOM Shows More Information

2D SOMs have boundaries, so cannot easily show the relationships between data mapped to the borders. An example is shown in Figure 3. We generated an artificial data set which containing 7 clusters in three-dimensional space. Each cluster contains 500 normally distributed points. Figures 3(b) and 3(c) are, respectively, the corresponding GeoSOM and 2D SOM. As can be seen, clusters 5 and 4 are close to each other in the original space. However, on the 2D SOM, they are mapped to opposite borders, which makes them appear to be far apart. The GeoSOM clearly shows that the two clusters are adjacent.

3.3 Graph Drawing on the Sphere

Compared to a 2D plane, a spherical surface provides a natural fisheye effect which enlarges the focus point and shows other portions of the image with successively less detail. The fisheye effect is a popular technique for visualizing large graphs(Herman, Melançon & Marshall 2000). As pointed out in (Meyer 1998), real spherical 3D-layout that allows interactive rotation opens a novel interaction for graph navigation. In (Kobourov & Wamplery 2004), the 2D spring-embedder was extended to layout graphs on the sphere. In this paper, we use a spherical circular layout to fine tune the nodes' positions. Details are shown in the next section.

4 Details of Our Approach

A relative small data set is used to demonstrate our approach — the import and export of manufactured metal between 32 countries in the year

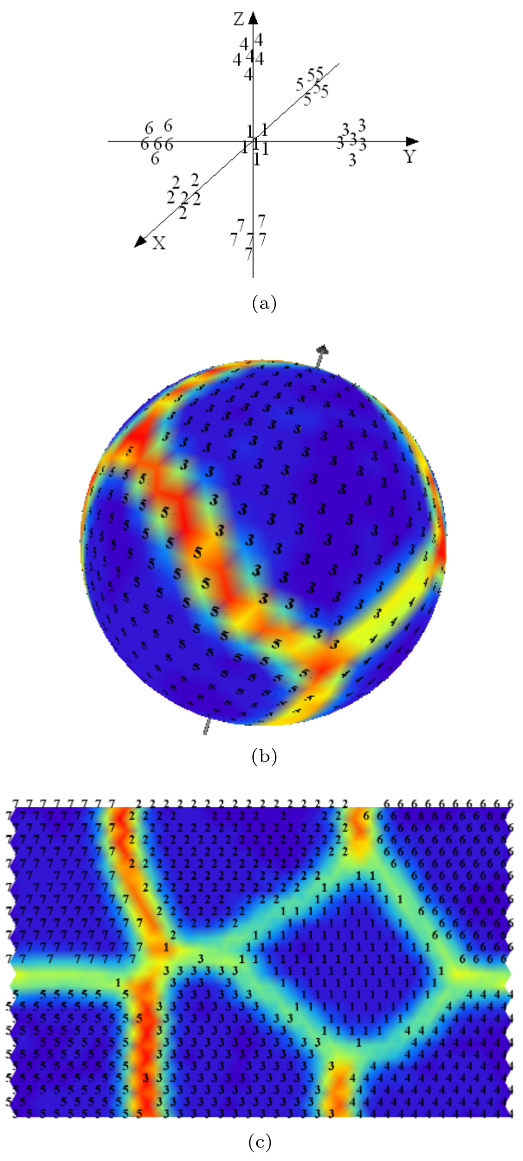


Figure 3: (a)The data set contains 7 clusters; (b) The GeoSOM; (c)The 2D SOM.

1994. It is extracted from the Pajek (Nooy, Mrvar & Batagelj 2005) database. Each country has four attributes: GDP, GDP growth, population and population growth. There are 62 edges in total. The edges are directed, pointing from exporting countries to importing countries.

There are two stages in our approach to visualize the multivariate network. The first stage is dimensional reduction using GeoSOM. In this stage, nodes close in high-dimensional space are mapped to similar areas on the sphere. Several nodes might be mapped to the same neuron (the same point on the sphere). So, in the second stage, they are arranged on a circle surrounding the neuron. For each node, our algorithm try to find a position on the circle that minimizes the number of edge crossings and avoids node-node and node-edge overlap.

4.1 The First Stage — Training of the GeoSOM

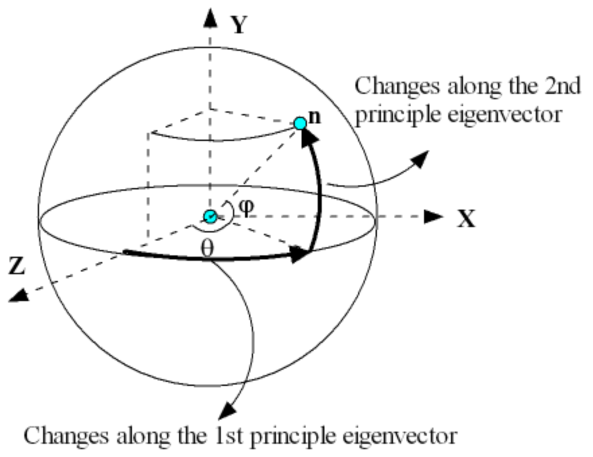
4.1.1 Linear Initialization

The GeoSOM use a two frequency icosahedron as the lattice which contain 42 neurons. Instead of random initialization, we first calculate the covariance matrix of the data set and then use its two principle eigenvectors to initialize the neurons. This is called linear initialization. It "has the effect of stretching the SOM to the same orientation as the data having most significant amounts of energy"(Hollmén 1996). So the training result are more stable and the SOM converges faster than random initialization.

The original linear initialization method is designed for the rectangular array: the two eigenvectors are arranged along the X and Y axis of the grid. Some modifications are needed in our case. The eigenvector with the largest eigenvalue is arranged along the parallels while the next eigenvector is arranged along the meridian. So the weight vectors are initialized as:

$$\vec{w}_i = \vec{m} + \alpha_1 \theta R \cos\varphi \vec{e}_1 + \alpha_2 \varphi R \vec{e}_2 \tag{5}$$

Here, $\vec{m}$ is the mean of the data set; α_1 and α_2 are two constants controlling the amount of change along the eigenvectors $\vec{e}_1$ and $\vec{e}_2$; R is the radius of the sphere which is one in our program; θ and φ are the longitude and latitude of the neuron (see Figure 4).



Y
X
Z
n
φ
θ
Changes along the 2nd principle eigenvector
Changes along the 1st principle eigenvector

Figure 4: Linear initialization of the GeoSOM.

4.1.2 Batch Mode Training

The training rule introduced in section 2 is called incremental training. The number of updating steps typically needs to be at least 500 times the number of neurons for the SOM to converge. There is a batch version of the SOM which runs much faster than the incremental version and the training results are approximately the same. In each updating step, every neuron collect a list of input data which are closest to its weight vector. Then the neuron's weight vector is set to be the mean over the data mapped to it and its neighborhood:

$$\vec{w}_j = \frac{\sum_{i=1}^{N} h_{b_i j}(t)\vec{x}_i}{\sum_{i=1}^{N} h_{b_i j}(t)} \tag{6}$$

where b_i is the best matching neuron of input $\vec{x}_i$ and N is the total number of inputs mapped to the neuron j and its neighbors; in our program, $h_{b_i j}$ is chosen to be a Gaussian function:

$$h_{b_i j}(t) = \exp\left(-\frac{\|\vec{r}_i - \vec{r}_j\|}{2\sigma(t)^2}\right) \tag{7}$$

where $\sigma(t)$ is the update radius decreasing linearly with time, and $\vec{r}_i$ and $\vec{r}_j$ are the position vectors of two neurons.

From a graph drawing point of view, nodes which are connected should be close to each other. So we modified the batch training rule by adding a constant β:

$$\vec{w}_j = \frac{\sum_{i=1}^{N} \beta h_{b_{ij}}(t)\vec{x}_i}{\sum_{i=1}^{N} h_{b_{ij}}(t)} \tag{8}$$

$\beta = 1$ if there are edges connecting the data mapped to neuron i and neuron j; $\beta = 0.85$ if the data are not connected. This has the effect of dragging together the data which are connected. Figure 5 shows the GeoSOM trained for 30 epochs with the world trade data set. In this visualization, the average distances between each neuron and its direct neighbors' weight vectors were calculated and coded with colors. The colors are linearly changing from blue, cyan, yellow to orange. Blue denotes the smallest variance between the weight vectors, and orange the largest. So data mapped to the same blue area are close in high-dimensional space and can be considered to lie in the same cluster. Each edge is drawn as a geodesic arc. A geodesic arc is the shorter part of the great circle that goes through the two nodes. It is the shortest path connecting any two points on the sphere.

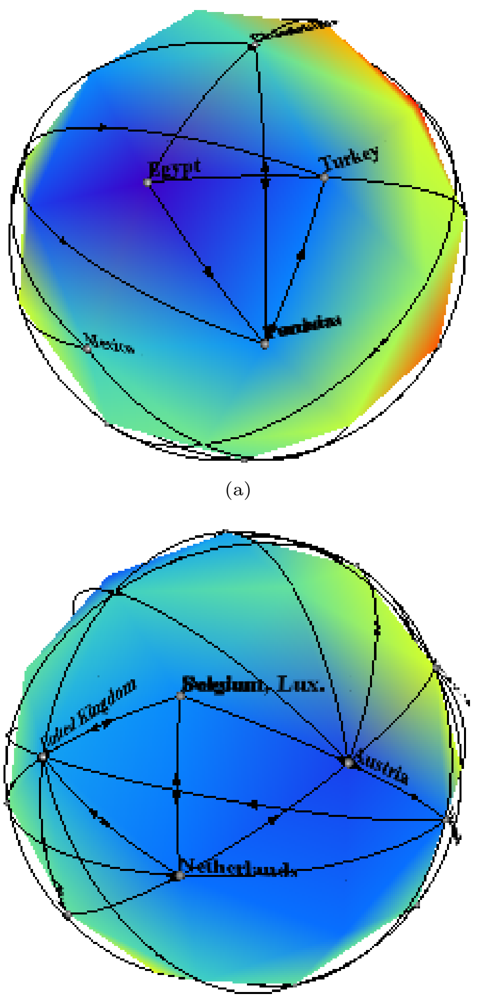


Figure 5: Two views of the GeoSOM trained with the world traded data set for 30 epochs.

4.2 The Second Stage — Circular Layout on The Sphere

As can be seen from Figure 5, some neurons have more than one inputs mapped to them. For example, Panama, Tunisia and Peru are projected to the same neuron. These graph nodes should be separated, but should not be placed too far away from their best matching neuron or the topology of the data map would be destroyed. Also, moving these nodes too far apart would prevent the use of colors to indicate how adjacent data are related and hence show the cluster boundaries.

We arrange the nodes on a circle surrounding the neuron. The radius of the circle is set to be one-third of the average distance between two neurons on the sphere. On each circle, we generate several equally separated positions. The number of positions is set to be three times the number of nodes the neuron collects. Each node can be placed on one of these positions or remains at the position of its best matching neuron (see Figure 6).

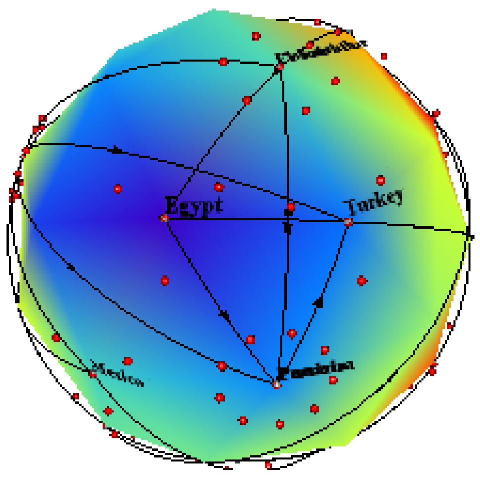


Figure 6: The small red spheres shows the positions where nodes at each neurons can be placed.

The algorithm try to minimize the number of edge crossings and avoid node-node/node-edge overlap when choosing the positions for the nodes. The node-node overlap is solved by not allowing any two nodes to be placed on the same position. The crossing minimization problem is NP complete(Gary & Johnson 1983). Currently, we tackle the problem by greedy search: place a node at each of the positions; calculate a cost function which sums up the number of edge crossings and the number of node-edge overlaps; place the node at a position with the lowest cost. This process is run for several iterations until the nodes' positions are relatively stable.

4.2.1 Detect Edge Crossings on the Sphere

On the 2D plane it is easy to detect whether two line segments intersect because they can be expressed as linear equations. Two steps are needed: calculate the intersection point of two straight lines and then test whether the point is inside both of the line segments.

Calculating the intersection point of two edges on the sphere is more complicated. Remember that an edge is part of the great circle which is the intersection between a sphere and a plane that goes through the center of the sphere. In our program, the center of the sphere is at the origin. Using Euclidean coordinates, the equation for a great circle is:

$$\begin{cases} x^2 + y^2 + z^2 = R^2 \\ ax + by + cz = 0 \end{cases} \tag{9}$$

Using spherical coordinates, the equations for a great circle are second ordered partial differential

Algorithm 1 Circular layout on the sphere

```
1: for each neuron do
2:     Calculate the equal distance positions;
3: end for
4: $m = 1$;
5: repeat
6:     for each neuron do
7:         for each nodes $n$ the neuron collects do
8:             $cost = \infty$;
9:             $pos = null$
10:            for each unoccupied position $p$ do
11:                Find the number of edge crossings $c_1$;
12:                Find the number of node-edge overlap $c_2$;
13:                $c = c1 + c2$;
14:                if $c < cost$ then
15:                    $cost = c$
16:                    $pos = p$;
17:                end if
18:            end for
19:            $n.pos = pos$;
20:            Mark $pos$ occupied;
21:        end for
22:    end for
23:    $m = m + 1$;
24: until $m >$ predefined running times
```

equations(Kreyszig 1968):

$$\begin{cases} \frac{d^2\theta}{dt^2} + \frac{2\cos\phi}{\sin\phi}\frac{d\theta}{dt}\frac{d\phi}{dt} = 0 \\ \frac{d^2\phi}{dt^2} - \sin\phi\cos\phi(\frac{d\theta}{dt})^2 = 0 \end{cases} \tag{10}$$

Although the above equations can be solved by numerical methods, the computation cost is too expensive for our purpose.

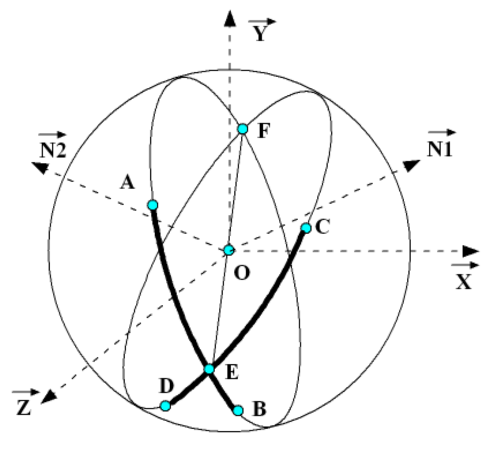


Figure 7: Calculate the intersection point of two edges on the sphere.

Fortunately, the intersection point of two great circles can be obtained geometrically as shown in Figure 7. Given two edges AB and CD, we first calculate the normals, $\vec{N_1}$ of plane AOB and $\vec{N_2}$ of the plane COD:

$$\vec{N_1} = \vec{OA} \times \vec{OB}, \tag{11}$$

$$\vec{N_2} = \vec{OD} \times \vec{OC} \tag{12}$$

Planes AOB and COD intersect at a straight line which in turn intersects the sphere at two points, E and F. They are the intersection points of the two great circles going through, respectively, AB and CD. The position vector of E and F can be calculated as:

$$\vec{OE} = \frac{\vec{N_1} \times \vec{N_2}}{\|\vec{N_1} \times \vec{N_2}\|}, \tag{13}$$

$$\vec{OF} = -\vec{OE} \tag{14}$$

The next step is to checked whether E or F are contained in both of the two edges. Take point E for example. We calculate the middle point M of edge AB. If the geodesic arc length EM is less than half of the length of AB, then E is inside the edge (see Figure 8). Whether E is inside edge CD can be checked in a similar way.

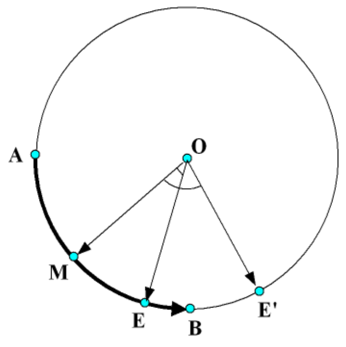


Figure 8: Check whether the intersection point E is inside edge AB.

4.2.2 Node-Edge Distance on the Sphere

A graph node should not be too close to edges which are not incident to the node; otherwise, the user could be confused about which are the edges' end points. In our program, the cost function will increase if the distance between a node and an edge is smaller than a threshold.

Given an edge AB and a node C on the sphere, we first need to find out a point P on the great circle G1 defined by AB such that the geodesic arc length of CP is the shortest. The point P can be obtained as follows (see Figure 9). First calculate the normal of plane AOB using equation (11). It intersect the sphere at point N. The great circle G2 defined by C and N is perpendicular to G1. In fact, P is one of the intersection point of G1 and G2. The coordinates of P can be calculated using the method introduced in the previous subsection.

If the geodesic arc length of CP is bigger than the threshold, then C is not too close to AB; otherwise, we need to check whether P is inside edge AB. If yes, then C is too close to AB and the cost function needs to be increased.

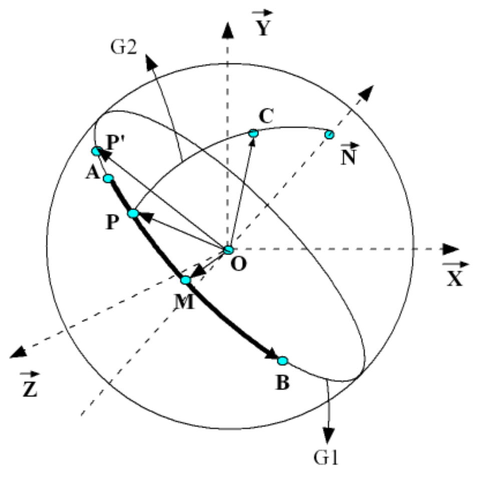


Figure 9: Distance between a node and an edge on the sphere

4.3 The Results

The program is run on a PC with a 3GHZ CPU and 1G of memory. For this small data set, SOM training took less than one second while the circular layout took 13 seconds. The final layout is shown in Figure 10. Compared to a normal graph layout, more trading patterns can be observed. Countries like Panama, Tunisia and Peru are close in the attributes space but there is no trading between them. On the other hand the European countries such as United Kingdom, Belgium and Netherland are major trading partners and also have very similar node attributes.

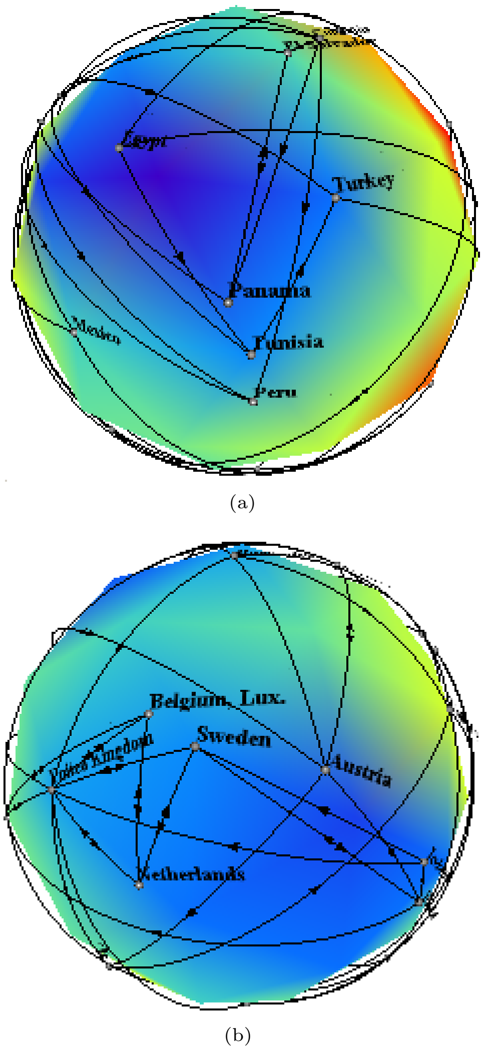


Figure 10: Final layout of the world trade network.

5 Two Dimensional Projection of the Spherical Image

The spherical image is not convenient for users to have a global view of the entire data set. Although they can rotate to see every part of the image, they have to remember the other half that a flat screen cannot simultaneously show. Hence, we implement an interface to project the spherical image onto a 2D plane. Currently, the Wagner III pseudocylindrical projection is used. This projection is neither equal-area or conformal. However, it has less extreme distortions and therefore can give a more balanced representation of the sphere's main features(Canters & Decleir 1989).

Figure 11 illustrates how the interface works. The user needs to select two points on the sphere. The first point A will be the center of the 2D projection. The second point B together with A defines a plane going through the center of the sphere. Suppose $\vec{OC}$ is the plane normal, then the sphere's central axis can be obtained by the cross product of the plane normals $\vec{OC}$ and $\vec{OA}$. This axis intersect the sphere at two points, N and S, which are made the "north" and "south" poles of the sphere. The geodesic arc going through these two points and opposite to A becomes the split line to open up the sphere.

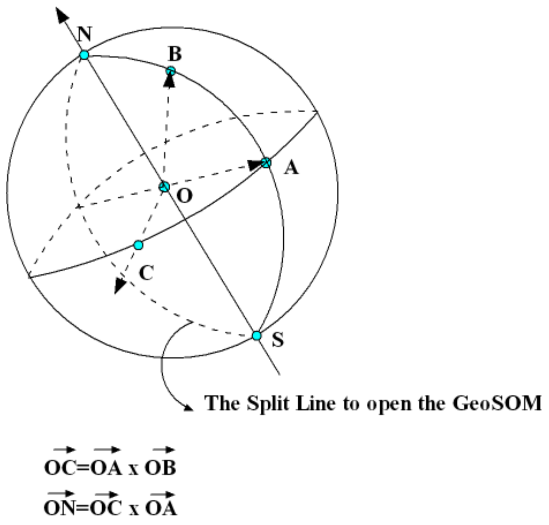

N
B
A
O
C
S
The Split Line to open the GeoSOM
$\vec{OC}=\vec{OA} \times \vec{OB}$
$\vec{ON}=\vec{OC} \times \vec{OA}$

Figure 11: Select two point on the sphere. The first point A will be the center of the projection. The second point B and A defined the central axis and the split line to open the sphere.

The Wagner III pseudocylindrical projection is not conformal. It means the angle between two edges will be changed after projection. This will affect the node-edge distance on the 2D image and may lead to new node-edge overlap. In order to preserve the original distance, for each edge, we transform the whole geodesic arc into 2D. So the edges on the 2D images are curves instead of straight lines. Figure 12 shows the 2D projection using Mexico as the image center. Edges cross the split line will be cut into two segments and drawn separately.

6 Conclusion And Future Work

We introduced a method to visualize the multivariate network. A spherical SOM (GeoSOM) was used to visualize the high-dimensional information contained in the graph nodes and produce the initial layout. We then implement a circular layout algorithm to fine-tune the nodes' positions, which tried to minimize the number of edge crossings and avoid node-node/node-edge overlaps.

However, our approach is only an initial attempt. There are many aspects that need to be improved — in particular, the circular layout. Basically, the problem is graph drawing with the geometric restriction that nodes cannot be placed too far away from the position of their best matching neurons. Currently, we restrict the nodes to lie on a circle. This maybe too restrictive for a good graph layout. Also, the greedy searching approach to find the right position for each node is time consuming. At each allowable position, we need to compute the crossing numbers and calculate the distance between each node to every other edge. The computational complexity is $O(nm + m^2)$, where n is the number of nodes and m is the number

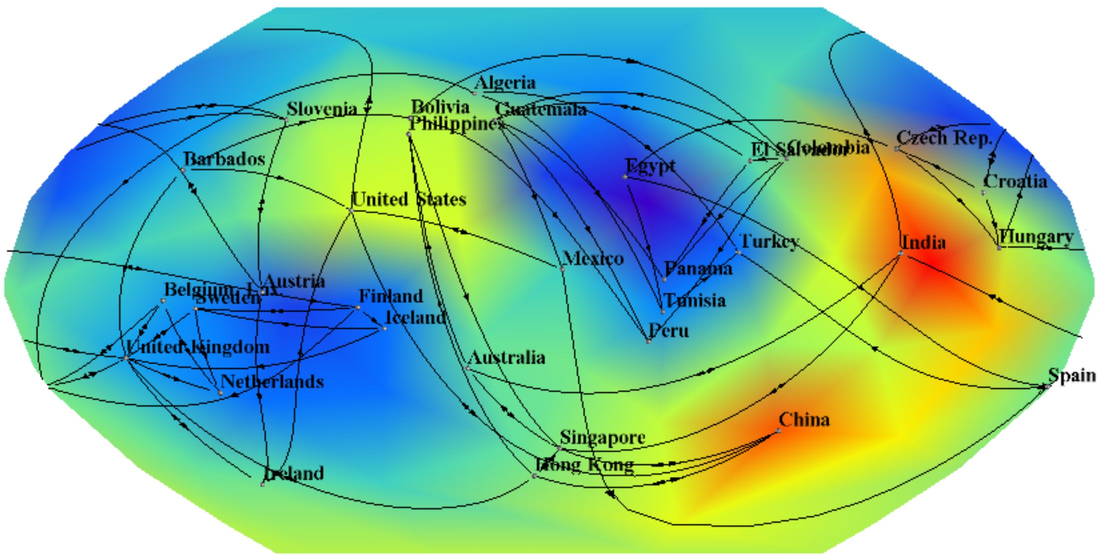


Figure 12: Project the spherical image on to 2D plane such that user can have a global view of the data set. Mexico is chosen to be the center of projection.

of edges. This would be too slow for large networks. We are looking into other more efficient combinatorial methods. Finally, for large networks, it is impossible to show all the nodes and edges; some abstraction methods, such as level-of-detail zooming, are needed.

References

Canters, F. & Decleir, H. (1989), *The World In Perspective: A Directory of World Map Projections*, John Wiley & Sons Ltd., England.

Eades, P. (1984), ‘A heuristic for graph drawing’, *Congressus Numerantium* **42**, 149 – 160.

Gary, M. R. & Johnson, D. S. (1983), ‘Crossing number is NP-complete’, *SIAM J. Algeraic and Discrete Methods* **4**, 312 – 316.

Herman, I., Melançon, G. & Marshall, M. S. (2000), ‘Graph visualization and navigation in information visualization: A survey’, *IEEE Transactions on Visualization and Computer Graphics* **6**, 24 – 43.

Hollmén, J. (1996), Process modeling using the self-organizing map, Master’s thesis, Department of Computer Science, Helsinki University of Technology.

Hotelling, H. (1933), ‘Analysis of a complex of statistical variables into principal’, *Journal of Educational Psychology* **24**, 417 – 441, 498 – 520.

Kamada, T. & Kawai, S. (1989), ‘An algorithm for drawing general undirected graphs’, *Information Processing Letters* **31**, 7 – 15.

Kobourov, S. G. & Wamplery, K. (2004), Non-euclidean spring embedders, *in* T. Munzer & S. North, eds, ‘Proceeding of The IEEE Symposium On Information Visualization 2004 (Infovis2004)’.

Kohonen, T. (2001), *Self-Organizing Maps*, Springer Series in Information Sciences, 3rd edn, Springer-Verlag, Germany.

Kreyszig, E. (1968), *Introduction to Differential Geometry and Riemannian Geometry*, University of Toronto Press.

Meyer, B. (1998), Self-organizing graphs - a neural network perspective of graph layout, *in* S. Whitesides, ed., ‘Proceedings of the 6th International Symposium on Graph Drawing’, Springer-Verlag, London, UK, pp. 246 – 262.

Nooy, W., Mrvar, A. & Batagelj, V. (2005), *Exploratory Social Network Analysis with Pajek*, CAMBRIDGE UNIVERSITY PRESS, 40 West 20th Street, New York, NY 10011-4211, USA.

Ritter, H. (1991), ‘Asymptotic level density for a class of vector quantization processes’, *IEEE Transactions on Neural Networks* **2**, 173 – 175.

Sammon, J. W. (1969), ‘A nonlinear mapping for data structure analysis’, *IEEE Transactions on Computers* **18**, 401 – 409.

Swayne, D. F., Buja, A. & Lang, D. T. (2003), Exploratory visual analysis of graphs in GGobi, *in* ‘Third Annual Workshop on Distributed Statistical Computing (DSC 2003)’, Viena.

Vesanto, J. (2002), Data Exploration Process Based on the Self.Organizing Map, PhD thesis, Department of Computer Science and Engineering, Helsinki University of Technology.

Wu, Y. & Takatsuka, M. (2005), The geodesic self-organizing map and its error analysis, *in* V. Estivill-Castro, ed., ‘Proceedings of the 28th Australasian Computer Science Conference (ACSC 2005)’, Australian Computer Society, Inc, pp. 343 – 351.