# GRAPH DRAWING IN SPHERICAL GEOMETRY

By

SCOTT MICHAEL PERRY JR

---

A Thesis Submitted to The Honors College

In Partial Fulfillment of the Bachelors degree
With Honors in

Computer Science

THE UNIVERSITY OF ARIZONA

M A Y 2 0 1 9

Approved by:

---

Dr. Stephen Kobourov
Department of Computer Science

# Graph Drawing in Spherical Geometry

Scott Perry

Dept. of Computer Science, Univ. of Arizona, Tucson, AZ, USA

**Abstract.** Graphs, or networks, are frequently visualized in two dimensions. However, some graphs do not have an ideal representation in two dimensions due to their inherent structure. In this paper, we discuss two distinct approaches for visualizing graphs in spherical geometry. We consider a projection-based method reliant on mathematics often seen in cartography, and present a working browser-based implementation of these ideas which exists as an additional feature of the GMap graph drawing tool. We then consider spherical multidimensional scaling as an alternative, which is commonly used as a dimensionality reduction technique. We explore its effectiveness as a graph embedder in spherical space, and present a series of graph drawings illustrating a working implementation.

## 1 Introduction

Graphs, or networks, are data structures that contain a set of data points which are related to each other through a set of edges that connect two single points, or nodes. Force directed algorithms, otherwise known as spring embedders, employ forces from the physical world to compute drawings of these graphs. The resulting graph drawings often contain symmetries and avoid crossing edges which makes them useful for visualizing the relationships in the underlying data. While force directed graphs have typically been reserved for Euclidean geometries, some datasets may be best observed on other geometries that have not been explored as extensively in this application. For example, while polyhedral graphs can be rendered in two dimensions without edge crossings, the edge lengths do not accurately represent the implied graph shape. However, in three dimensions, both properties can be preserved. To identify the practicality of rendering graphs directly on the sphere, we attempted two separate approaches. First, we projected layouts produced by Euclidean spring embedders to spherical geometry. Then, we applied the dimensionality reduction technique, multidimensional scaling (MDS) [7], to graph drawing with the constraint that points must reside on the sphere.

Due to the nature of spherical geometry relative to Euclidean geometry, drawing force directed graphs on a sphere requires handling properties specific to the sphere. For example, there is not a unique shortest distance between any two points, and distance between two points can be interpreted as either Euclidean distance or geodesic distance. Because of the geometric assumptions we make with common force directed drawing methods, which are explained in detail in [15] no longer hold, applying forces to nodes on a sphere is a unique task. In our first approach, we interpreted our problem as one of translating geometric spaces rather than altering how the forces were applied. Rather than perform force calculations on the sphere, the layouts were produced in two dimensional Euclidean space, projected onto a sphere, and reprojected back onto the plane to display the layout in a web browser.

In this work, we describe the details related to inversely projecting a two dimensional plane onto a sphere, as well as applying another projection function to transfer the drawing

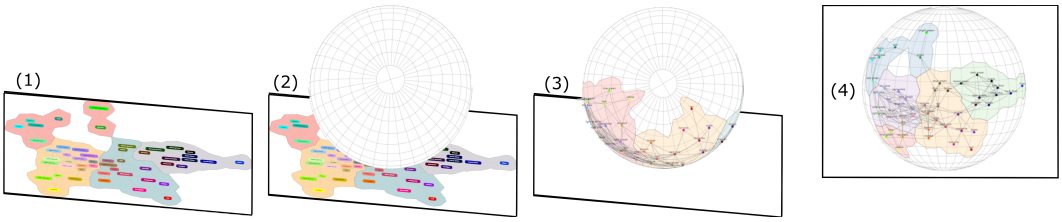



Fig. 1: Rendering of a sample GMap force directed graph inversely projected onto the sphere, and then orthographically projected into the browser. Step 1 of the pipeline depicts a standard drawing from GMap. Step 2 lays a sphere over the initial GMap drawing. During step 3, the GMap drawing is inversely projected onto the sphere. The drawing is then orthographically projected onto the browser in step 4.

from the sphere back into the plane. In section 2 we discuss related works on spherical graph drawing. Section 3 reviews properties of sphere to plane projection formulas in order to determine the optimal choice for an inversion function as well as for reprojecting the sphere onto the plane. This leads to a more detailed discussion in section 4 about rendering graphs on a browser with labeled nodes and the ability to zoom. Section 5 provides results from a few example graphs, and section 6 elaborates on our alternate approach to spherical embeddings, which relies on MDS. Section 7 summarizes our findings and outlines plans for future work.

The spherical force directed graph layout implementation is built as a new visualization option of the existing graph visualization tool GMap, which uses geographic-like maps to display graphs by clustering nearby nodes based on their relationship into geographic entities [11]. A working example is available at `http://gmap.cs.arizona.edu`.

## 2 Related Work

While few have published papers on drawing graphs in spherical space, those that have used a variety of techniques. The paper [4] is one of the oldest in literature that proposes a solution, applying MDS in a manner similar to that described by Kruskal in [18]. Kruskal was one of the first to publish about non-metric multidimensional scaling. In [18], the stress function

$$S = \sqrt{\frac{\sum_{i<j}(d_{ij} - \hat{d}_{ij})^2}{\sum_{i<j} d_{ij}^2}}$$

models an ideal configuration of a drawing that associates distances in the drawing with dissimilarities in the data. The variable $d_{ij}$ contains the initial distance between points i and j, while $\hat{d_{ij}}$ holds the distance between points i and j that keep the same rank order as the dissimilarities between points i and j. In our case, dissimilarities are measured as the shortest path between nodes on the graph. If the dissimilarity of points i and j is stored as $\delta_{ij}$, then if $\delta_{ij} \leq \delta_{mn}$ then $\hat{d}_{ij} \leq \hat{d}_{mn}$. This method is described as nonmetric because it relies on the ordering of the distances rather than the distances themselves. Common methods to minimize the stress function include various forms of gradient descent as further outlined in [19]. In summary, gradient descent for MDS begins with an initial configuration for $\hat{d}_{ij}$

which represents the points in the visualization space. Once the gradient of $S$ is calculated, which describes the direction that $S$ increases the greatest, we add the negative gradient to each $\hat{d}_{ij}$. We repeat this process until we find a suitable convergence, however converging to the global minimum is not guaranteed given our stress function.

Because the stress function above relies on preserving the rank order of dissimilarities in the distances, the paper [4] argues that rather than using spherical arc distance in the formula, Euclidean distance between points on the surface of the sphere can be substituted, producing an equally suitable configuration. The equation for $d_{ij}$ from [4]is as follows:

$$d_{ij} = \{2 - 2\sin\theta_{i2}\sin\theta_{j2}\cos(\theta_{i1} - \theta_{j1}) - 2\cos\theta_{i2}\cos\theta_{j2}\}^{\frac{1}{2}}$$

where $\theta_{i1}$ is the azimuthal angle for point $i$ and $\theta_{i2}$ is the zenith angle for point $i$. Euclidean distances are suitable approximations for the geodesic distances due to the "one to one increasing relationship between the Euclidean distance and the arc length" [4]. Using this assumption, Cox provides partial derivatives for the equation described in Kruskal which is then minimized with gradient descent.

The paper [27] solves a similar problem to the one we pose, finding an ideal visualization given data that is related in one dimension but has many other dimensions as well. For example, countries may be related through trade, however they also have other information such as population, GDP, etc. To visualize the dimensions with a single drawing, [27] suggests applying a self-organizing map (SOM) to the data with point initialization based on principal component analysis (PCA). Note that while principal component analysis is a dimensionality reduction technique, it can not be directly applied to the stress function defined by [4] because it requires linearity which is not present on the sphere.

A different approach to laying out graphs on a sphere is described in [16], where force directed algorithms are applied in Riemannian geometry. During a traditional force directed algorithm, the forces between each node and every other node are calculated. Thus, we can assess a single node at a time, determining a single force vector that represents the sum of all other nodes' forces upon the one being assessed. In order to translate the vectors in Riemannian geometry [16] relies on tangent planes. On a sphere, a tangent plane exists for every point, and when a node is being assessed, every other node can be mapped onto the tangent plane of the node under inspection. Once the nodes are on the tangent plane, applying forces to them is a similar task to that of Euclidean geometry. Then, the nodes can be translated back onto the sphere, and the iterative force directed algorithm continues.

## 3 Sphere to Plane Projections

Spheres do not have a representation on a plane that perfectly preserves direction, area, shape and distance. Much like an orange peel cannot be cut so that it is perfectly flat, spheres do not have an exact parallel in two dimensional Euclidean space. Thus, any attempt to represent a sphere on a plane results in the loss of information, and can be classified as a projection. Various sphere to plain projections have been explored extensively in the field of cartography, and are the basis for two dimensional maps. Our approach to rendering a graph on the surface of a sphere can be divided into two separate tasks. The first is to model an existing graph in two dimensional Euclidean geometry with a sphere, and the second is to display that sphere in a browser. In order to explain our approach to each task, we will review the essentials of spherical projection formulas. For a more thorough explanation of the how spheres can be represented, see [14].

### 3.1 Projection Basics

Because projections cannot preserve direction, area, shape, and distance, each projection formula is created for a specific purpose. Projections that preserve area are considered equal area, while those that preserve distance are classified as equidistant. If the direction along some of the great circles on the sphere remain, then the projection is labeled as a true-direction one, and if local shapes are preserved, then it is labeled as conformal. Some of the most common projections first translate the sphere to another three dimensional figure that is able to be flattened more easily. Figures such as cylinders, cones, and planes can all form tangential contact points with a sphere. When translated, area, shape, distance, and direction at these contact points are not distorted. As points on the sphere become more distant from the other figure, the amount of distortion in at least one of these traits increases.

Each of the three dimensional figures used to translate the sphere to the plane produce uniquely shaped resulting images in the plane. Conic projections often capture a single hemisphere, and in the plane appear as an unraveled cone where the most accuracy occurs on a single parallel of the sphere. In planar projections, the plane that is projected to lies tangential to a single point on the sphere. Thus, the projection captures the relationship between other points on the sphere and the focus point. These projections can most accurately capture a single hemisphere. Cylindrical projections capture the entire sphere, by surrounding the sphere with the side of the cylinder along a single parallel. Often, the equator is used as the line of tangency, and in the Mercator projection which is the most common map projection, the meridians of the sphere are equally spaced, with the space between parallels increasing towards the poles.

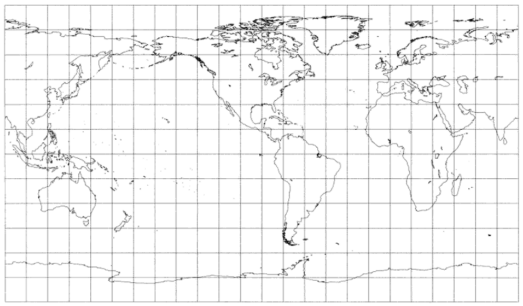


Fig. 2: Rendering of the Earth using the equirectangular projection. Distortion of shape and area clearly increase towards the poles. [24]

### 3.2 Plane to Sphere Inverse Equirectangular Projection

The equirectangular projection is a cylindrical projection which maps a sphere onto a cartesian grid that contains equal sized squares. Meridians are mapped to vertical lines and parallels to horizontal lines. The simplest form of this projection is the Plate Carrée, where the line of tangency between the cylinder with the sphere is the equator. Considering $\lambda$ is the longitude and $\phi$ is the latitude of the point on the sphere to project, the Plate Carée projection is described by the following trivial formula:

$$x = \lambda$$

$$y = \phi$$

The projection is not conformal nor equal area, however, a two dimensional force directed graph layout can be thought of as a collection of individual points. The distortion seen among shapes has lesser implications than maintaining proportionally similar distances between points. In the existing GMap software, the force directed algorithms produce rectangular layouts. Because cylindrical projections produce rectangular results, they are the optimal type to use for inverting an collection of points onto a sphere when the accuracy of the shapes is minimally important. The simplicity of the Plate Carée projection allows the two dimensional graph to be quickly translated onto a sphere even for large numbers of nodes.

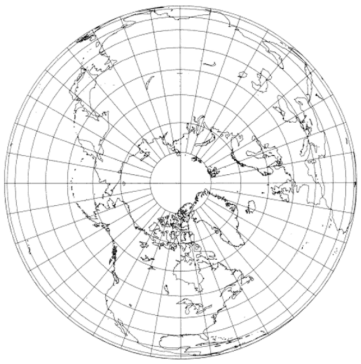


Fig. 3: Rendering of Earth through an orthographic projection where the rays used to calculate the Euclidean position of points were taken from infinite distance from the north pole. [24]

### 3.3 Sphere to Plane Browser Orthographic Projection

Once the force directed graph layout has a spherical representation, it can then be displayed in the browser. While cylindrical projections produce rectangular shaped layouts in a plane, planar projections result in circular visualizations. Because we aim to keep the aesthetic similar to that of a globe, the orthographic projection is a suitable choice. An orthographic projection translates the sphere to the plane by casting rays from infinite distance through the sphere that are orthogonal to the projection plane. The location on the surface of the sphere where the ray passes through is mapped to where that same ray hits the plane. When projecting an entire sphere, points will overlap because each of the rays cast pass through the sphere at two points. Thus, it is common to apply the orthographic projection to a single hemisphere.

According to [23], the mathematics of the orthographic projection is as follows:

$$x = R\cos\phi\sin(\lambda - \lambda_0)$$

$$y = R[\cos\phi_1 \sin\phi - \sin\phi_1 \cos\phi\cos(\lambda - \lambda_0)]$$

$$h' = \sin\phi_1 \sin\phi + \cos\phi_1 \cos\phi \cos(\lambda - \lambda_0)$$

$$k' = 1.0$$

"where $\phi_1$ and $\lambda_0$ are the latitude and longitude, respectively, of the center point and origin of the projection, $h'$ is the scale factor along a line radiating from the center, and $k'$ is the scale factor in a direction perpendicular to a line radiating from the center."

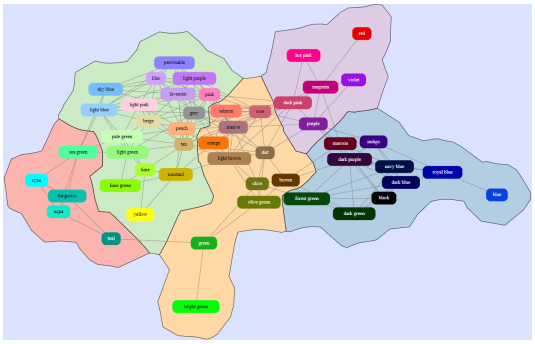


Fig. 4: Output from the GMap software using the embedding algorithm from [12], where each region seen is stored in an XDOT file [11]

## 4 Rendering Graphs in the Browser

The working example of the spherical graph projections at `http://gmap.cs.arizona.edu` relies on the JavaScript library D3.js [2] to handle the computations of the projections and rendering of the graphics which are created with scalable vector graphics (SVG). D3.js allows for binding of data to the document object model (DOM), which allows graphs to be efficiently displayed and manipulated. Conveniently, D3.js contains implementations of common projections, and other helpful spherical calculation formulas in its Geographies module.

In GMap, graphs are input in DOT format [17] which specifies a list of nodes, each containing an identifying field along with a list of attributes. Following the nodes, a list of edges is defined using the identifying fields of the nodes. Further details of the specification are outlined by [9]. The first step of GMap is to embed the graphs in two dimensional Euclidean space [11]. Current options include sfdp [12] and neato [13]. The former is a multi-level force directed algorithm which relies on the Barnes and Hut approximation algorithm to optimize long-distance force calculations. The latter approaches the graph drawing problem as an energy optimization problem where imaginary springs are placed between all the nodes of the graph, each with a desirable length. Thus, the optimal configuration for the graph is when all springs between nodes are at their desirable length. These lengths are a function of the graph's shortest path matrix, and therefore the stress function from [13] is similar to that in Multidimensional Scaling (MDS).

### 4.1 XDOT Parsing

Once the graph has been embedded in two dimensions, the nodes are clustered, and a map is generated from the clusters. Once these regions have been defined, they are stored in various visual file formats such as PNG, SVG, or XDOT. XDOT is a variation of DOT that includes positions for regions and nodes along with other relevant information such as coloring [10]. To use D3.js' functions for rendering data through spherical projections, the XDOT file must first be parsed to extrapolate the vertices of each region in the map. D3.js requests geographic data to be presented in GeoJSON format when performing projection calculations [3]. According to the GeoJSON specification, regions must be defined with vertices in counter-clockwise order. The following formula estimating the area under the region was applied to each set of vertices to determine the orientation of the region:

$$\text{orientation} = \sum_{i=0}^{n-1} (x_i - x_{i+1})(y_i + y_{i+1})$$

If the output of the summation for the region is negative, then the points are in counter-clockwise order. Otherwise, the order of the vertices must be flipped. Once the correctly ordered regions are defined, the vertices of each region are passed through the equirectangular inverse projection formula to map the regions to the sphere. Then, they are filtered through the ortographic projection formula to map them back onto the plane. Nodes and edges are handled in a similar manner, where a node is defined by a point, and edges are defined by two points. However, nodes and edges do not require the orientation preprocessing step. Instead, their geometric properties are encompassed by the GeoJSON tags, "Point" and "LineString".

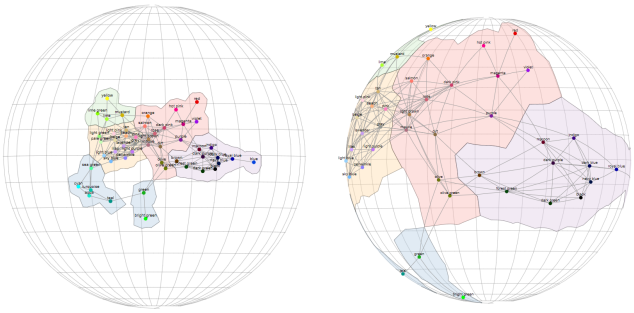
### 4.2 Selective Drawing

The two dimensional force directed graph is mapped to the entirety of the sphere, and the orthographic projection produces overlapped results. However, D3.js provides the means for clipping a visualization that extend beyond the boundary specified. To capture a single hemisphere, 90°is the appropriate parameter. When drawing labels, this built in function is no longer applicable, because text is drawn in a different manner than nodes, edges, and regions. Thus, our own clipping function must be implemented so that labels are drawn only for the nodes that are visible. Effectively, given a set of points, we are attempting to identify if each point exists on a hemisphere at a specific center that lies on the surface of the sphere.

We can identify the center of our desired hemisphere in a similar way that we inverted points from the two dimensional force directed graph embedding onto a sphere. Instead, we take the center point of the projection from the browser, and pass it through the same inverse equidistant projection formula used previously. Then, for each node's spherical coordinates, we calculate the distance from the node to the selected center using D3.js' built in function. For simplicity we use a unit sphere as the intermediate mapping step. Thus, if the distance from a node to the center point is greater than $\frac{\pi}{2}$, then neither the node, nor the label should not be drawn.

### 4.3 Coverage

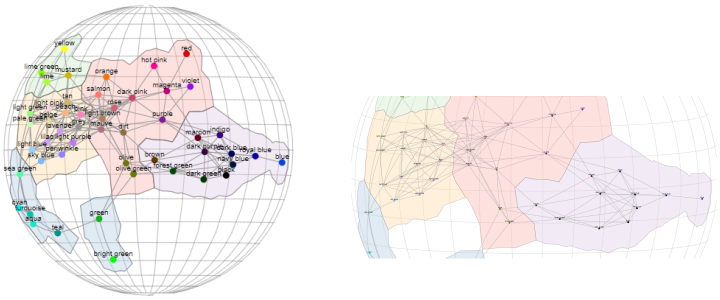
The coverage setting for spherical drawings in GMap corresponds to the inverse of the scale of the equirectangular projection. However, this scale represents the distance between



(a) Minimal Coverage (b) Increased Coverage

Fig. 5: As the coverage slider of GMap increases, the elements of the sphere are scaled, keeping the sphere at a constant size.

points of the output that an equirectangular projection would produce given a sphere. In our pipeline, the two dimensional force directed embedding is of fixed size, and is our input, rather than a sphere. Hence, increasing the scale of the projection makes the two dimensional graph cover less of the sphere when the two dimensional graph embedding is filtered through the inverse projection function. Thus, increasing the coverage slider in GMap decreases the scale of the projection and the graph covers more of the sphere.



(a) Minimal Zoom (b) Increased Zoom

Fig. 6: As the zoom slider of GMap increases, the sphere and contents are both scaled at a proportional rate, giving a zoom effect.

### 4.4 Zoom

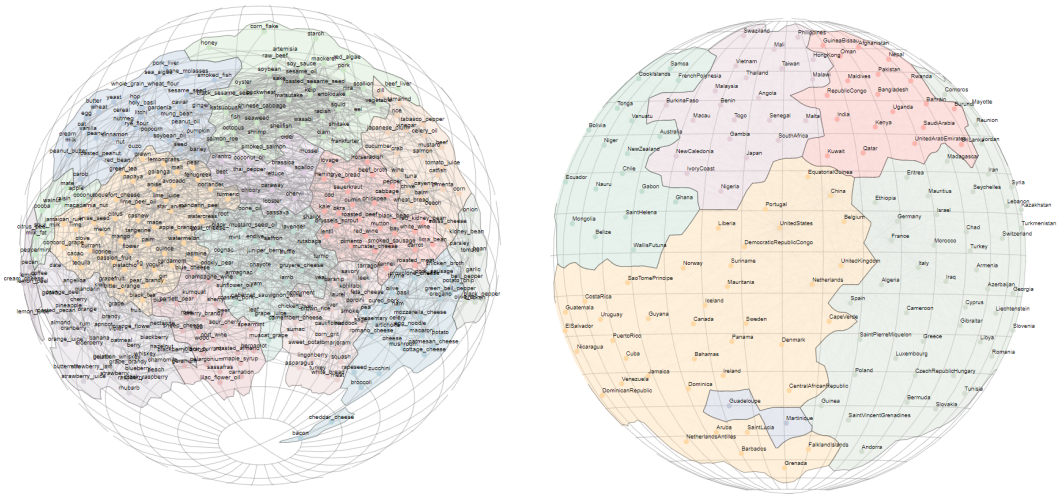
The zoom parameter corresponds directly to the scale of the orthographic projection. Increasing the scale of this projection increases the output size in the browser of every node, edge, and region linearly. By adjusting the zoom parameter, the user can effectively decide

the context in which they want to view the central data. When the zoom is greater, the user has a more isolated view of a specific region.

## 5 Spherical Layout Examples

The software is available for use at `http://gmap.cs.arizona.edu` by selecting the "Spherical" visualization type checkbox under the advanced options tab. While the layout and clustering algorithms are performed on the server, performance of spherical visualizations is dependent on the user's machine due to D3.js handling many of the calculations. Graph's under a few hundred nodes can be easily visualized with this tool, however larger datasets may become cluttered. This is partially due to the restricted domain of a sphere with a set radius [26] and to the space required for labels for each node. The former issue is partially remedied through the zoom functionality, however perspective of the entire sphere is lost. The latter problem requires semantic zoom functionality, so further analysis is needed to determine the most meaningful nodes given the current view.

Because the mapping functionality of GMap is based around the construction of Voronoi diagrams for a given set [11], the resulting two dimensional embedding is a single mass. When inversely projected onto a sphere and then reprojected back into the plane, the same mass exists. Paralleling a globe, our force directed drawings resemble a super-continent rather than the continents on a present-day globe. Yet, the connected nature of our graph drawings allows viewers to see the entire graph as a whole, or with selective context via the zooming functionality.



(a) Recipes (b) Trade

Fig. 7: Two examples of DOT graphs visualized through `http://gmap.cs.arizona.edu` where the Recipes graph depicts ingredients that share recipes while Trade maps trade partners to a sphere. Note that the edge opacity of the Trade visualization was reduced for clarity. Visualizations of countries in our world is a common domain to relate to spherical visualizations [21].

## 6 Multidimensional Scaling

Multidimensional Scaling (MDS) is a dimensionality reduction technique reliant on comparing the dissimilarities of data with the dissimilarities represented by a visualization configuration. The difference between the data's dissimilarities and the configuration's dissimilarities is modeled by a stress function dependent on the problem. Stress functions typically incorporate the distance metrics used in the data and in the visualization space. The variability of stress functions led to the development of a variety of MDS forms such as metric, nonmetric, and classical. The originally developed approach was designed by Torgerson, a psychometrician, in 1952 which required the psychological data to be estimated numerically [25], thus it is considered the first metric method. Other psychometricians such as Shepard and Kruskal allowed data to be defined ordinally rather than numerically ([22], [19]). By fitting an optimal configuration to the monotonically ordered data through isotonic regression, nonmetric MDS was created. While we will focus on metric MDS as a solution to our problem, the other forms are well described with clearly defined stress functions in [5].

### 6.1 MDS for Graph Drawing

Graph drawing can be modeled as an MDS problem as there are data dissimilarities and a desired visualization space. Given there are $N$ nodes in the graph, it can be represented by an $N \times N$ matrix $\delta$ where $\delta_{ij}$ is the number of edges in the shortest path from node $i$ to node $j$. This matrix $\delta$ will represent the data for our MDS problem. Our visualization is constrained by the intent of this paper as the surface of a sphere in $\mathbb{R}^3$. In [7], De Leeuw identifies the distance between nodes $i$ and $j$ in the visualization space as

$$d_{ij} = \lambda \arccos(\frac{x_i' x_j}{\lambda^2})$$

where $\lambda$ is the radius of the sphere and $x_i, x_j$ are the vectors representing nodes $i, j$ in the visualization space. Effectively, $d_{ij}$ can be interpreted as the great-circle distance between two points. Therefore, we can define the optimal configuration representing the graph as one that minimizes the summed difference between $\delta_{ij}$ and $d_{ij}$ for every pair of nodes. Formally specified by De Leeuw:

$$\text{stress} = \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij}(\delta_{ij} - d_{ij})^2$$

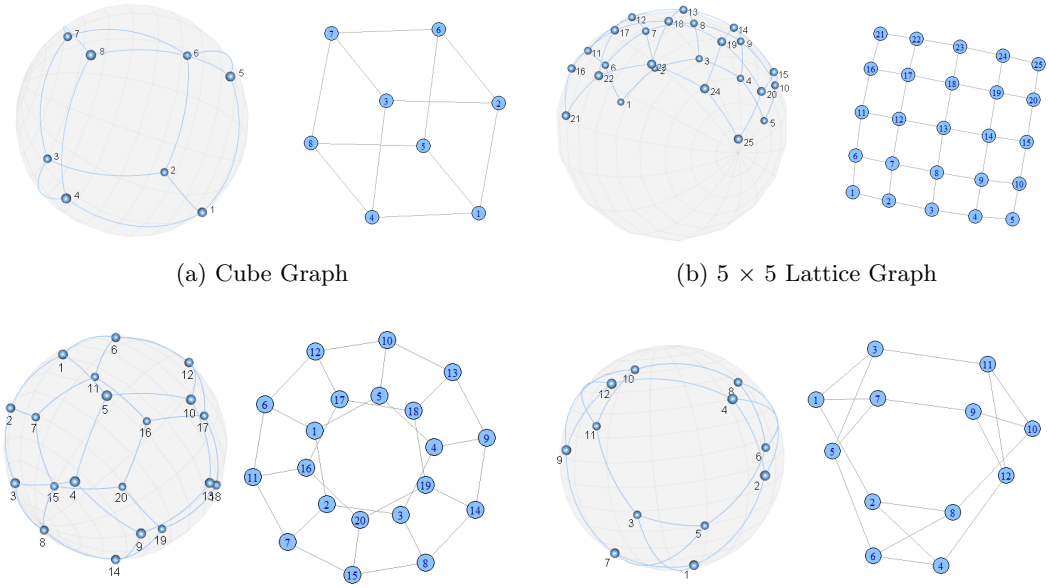
where $N$ is the number of nodes in the graph and $w$ is an $N \times N$ matrix of weights which for our implementation will be a matrix of ones.

While the resulting configuration defines points in $\mathbb{R}^3$, CMDA is an effective method to enforce the constraint that nodes are placed on the surface of a sphere in the visualization space [20]. In CMDA, a penalty parameter is assigned which penalizes configurations that do not form to the constraint. This can be modeled by the function:

$$\sigma_\kappa(X) = \min_{\Delta \epsilon D_L} \sigma(X, \Delta) + \kappa \min_{\Delta \epsilon D_C} \sigma(X, \Delta)$$

where $\Delta \epsilon D_L$ is the graph theoretic distance matrix and $\Delta \epsilon D_C$ contains the spherical constraints. The penalty parameter is $\kappa$ and the stress functions are $\sigma_L(X, \Delta)$ and $\sigma_C(X, \Delta)$ where $X$ is the configuration matrix for the point positions on the sphere in Euclidean

coordinates and $\Delta$ is the dissimilarity matrix for the stress function. When $\kappa = \infty$, the second term of $\sigma_\kappa(X)$ "is forced to zero, and we minimize the first term under the conditions that the second term is zero," meaning that all points lie on the sphere. As with the rest of the paper, De Leeuw identifies a way to optimize this adjusted stress function through majorization.



(a) Cube Graph

(b) $5 \times 5$ Lattice Graph

(c) Dodecahedral Graph

(d) Franklin Graph

Fig. 8: Common graphs from the package 'igraph' [6] processed through 'smacof' [8] and rendered onto the sphere with 'rgl' [1]. Note that some graphs such as the dodecahedral graph are unable to be accurately rendered in two dimensions, while graphs such as the lattice cause distortion to the graph as a whole, although each pair of points' distances are optimal.

### 6.2 Implementation and Examples

The package 'smacof' [8] is implemented in R with functionality for performing MDS with spherical constraints enforced by CMDA. To use this package, we must first develop our dissimilarity matrix for the graph. Under the assumption that our graph inputs are unweighted, we calculate the all pairs shortest path matrix using a breadth-first search approach. After processing this matrix through the 'smacof' package, we retain the optimal configuration in coordinates in $\mathbb{R}^3$. Because of the CMDA penalty adjustment, we assume all points in this configuration lie on the surface of the sphere. Knowing the configuration is centered at $\overrightarrow{0}$,

we can calculate the radius of the sphere with the formula:

$$\text{radius} = \sqrt{x^2 + y^2 + z^2} = \left\| \begin{bmatrix} x \\ y \\ z \end{bmatrix} \right\|^2$$

Using the OpenGL-based R rendering package 'rgl' [1] we render a sphere of the produced radius, and plot each node according to its point in the optimal configuration along with its corresponding label. Using a predefined set of graphs from the 'igraph' package [6], we render arcs where there exists an edge between two nodes. Note that the MDS approach calculates the nodes' positions on the domain of the sphere, thus the resulting graphs naturally occupy most of the sphere. Therefore, a coverage parameter is not necessary like it was in the projection-based approach.

## 7 Conclusions and Future Work

While the first method we introduced for creating spherical embeddings was reprojection, there are some notable properties of the output that have not yet been mentioned. Although the coverage parameter of our implementation attempts to wrap the graph drawing around the sphere, there are restrictions as to its effectiveness. The graph represented on the sphere is bound by the shape of initial two dimensional force directed drawing, thus a rectangular two dimensional drawing will not be able to cover the majority of the sphere without wrapping around. However, in the MDS approach, this is not an issue, as the calculations are performed iteratively in the spherical geometry. MDS converges when all distances between nodes in the visualization space equal their graph-theoretic distances, and the radius of the sphere is variable so that the drawing can cover the entire sphere.

Additionally, reprojecting onto a sphere loses the notion of distance from points on its border. In the two dimensional drawing, nodes in opposite corners are the furthest apart both visually and by their graph-theoretic distances. However, as the coverage of the drawing grows larger than a single hemisphere these nodes become the closest in the visualization. Therefore, the graph-theoretic distances in the visualization lose accuracy as it grows in size. Again, MDS does not suffer the same inaccuracy for the same reason that it avoids the previous property. For these reasons, we believe that spherical MDS better represents graphs on the sphere.

Because De Leeuw has outlined a way to optimize the MDS stress function through majorization in [7], we aim to develop another visualization option in GMap that utilizes the 'smacof' package. While we have explored the practicality of visualizing graphs on the sphere, we will consider other quadratic surfaces as hosts for our graphs.

## References

1. Daniel Adler, Duncan Murdoch, and others. *rgl: 3D Visualization Using OpenGL*, 2019. R package version 0.100.19.
2. Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. $D^3$ data-driven documents. *IEEE transactions on visualization and computer graphics*, 17(12):2301–2309, 2011.
3. Howard Butler, Martin Daly, Allan Doyle, Sean Gillies, Stefan Hagen, and Tim Schaub. The geojson format. Technical report, 2016.

4. Trevor F Cox and Michael AA Cox. Multidimensional scaling on a sphere. *Communications in Statistics-Theory and Methods*, 20(9):2943–2953, 1991.
5. Trevor F Cox and Michael AA Cox. *Multidimensional scaling*. Chapman and hall/CRC, 2000.
6. Gabor Csardi and Tamas Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695, 2006.
7. Jan De Leeuw and Patrick Mair. Multidimensional scaling using majorization: Smacof in r. 2011.
8. Jan de Leeuw, Patrick Mair, and Maintainer Jan de Leeuw. Package 'smacof', 2013.
9. Emden R Gansner and John Ellson. The dot language, 2002.
10. Emden R Gansner and John Ellson. Output formats, 2002.
11. Emden R Gansner, Yifan Hu, and Stephen Kobourov. Gmap: Visualizing graphs and clusters as maps. In *2010 IEEE Pacific Visualization Symposium (PacificVis)*, pages 201–208. IEEE, 2010.
12. Yifan Hu. Efficient, high-quality force-directed graph drawing. *Mathematica Journal*, 10(1):37–71, 2005.
13. Tomihisa Kamada, Satoru Kawai, et al. An algorithm for drawing general undirected graphs. *Information processing letters*, 31(1):7–15, 1989.
14. Melita Kennedy, Steve Kopp, et al. *Understanding map projections*. ESRI, 2000.
15. Stephen G Kobourov. Force-directed drawing algorithms. 2004.
16. Stephen G Kobourov and Kevin Wampler. Non-eeuclidean spring embedders. *IEEE Transactions on Visualization and Computer Graphics*, 11(6):757–767, 2005.
17. Eleftherios Koutsofios, Stephen North, et al. Drawing graphs with dot. Technical report, Technical Report 910904-59113-08TM, AT&T Bell Laboratories, Murray Hill, NJ, 1991.
18. Joseph B Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964.
19. Joseph B Kruskal. Nonmetric multidimensional scaling: a numerical method. *Psychometrika*, 29(2):115–129, 1964.
20. James Charles Lingoes and Ingwer Borg. *Geometric representations of relational data: Readings in multidimensional scaling*. Mathesis Press, 1979.
21. Tamara Munzner. H3: Laying out large directed graphs in 3d hyperbolic space. In *Proceedings of VIZ'97: Visualization Conference, Information Visualization Symposium and Parallel Rendering Symposium*, pages 2–10. IEEE, 1997.
22. Roger N Shepard. Multidimensional scaling, tree-fitting, and clustering. *Science*, 210(4468):390–398, 1980.
23. John Parr Snyder. *Map projections–A working manual*, volume 1395. US Government Printing Office, 1987.
24. John Parr Snyder and Philip M Voxland. *An album of map projections*. Number 1453. US Government Printing Office, 1989.
25. Warren S Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, 17(4):401–419, 1952.
26. Glen Van Brummelen. *Heavenly mathematics: The forgotten art of spherical trigonometry*. Princeton University Press, 2012.
27. Yingxin Wu and Masahiro Takatsuka. Visualizing multivariate network on the surface of a sphere. In *Proceedings of the 2006 Asia-Pacific Symposium on Information Visualisation-Volume 60*, pages 77–83. Australian Computer Society, Inc., 2006.