**Lines:** 400

# Browser-based Hyperbolic Visualization

Stephen Kobourov and Jacob Miller

Department of Computer Science, University of Arizona

**Abstract.** While most graphs are drawn in the 2-dimensional Euclidean plane, some properties of non-Euclidean geometries may be beneficial for graph visualization. Some graphs can be better embedded in hyperbolic space than in Euclidean space and hyperbolic geometry provides natural "focus+context," with parts of the graph near the center of the view shown large and those far from the center progressively smaller. We consider two methods for hyperbolic visualization of graphs in the browser. The first makes use of hyperbolic analogues to spherical projections. We integrate it with the GMap system to provide GMaps, MapSets, BubbleSets, and LineSets, as well as traditional node-link diagrams. The second method relies on a generalization of force-directed algorithms to non-Euclidean geometries, making better use of the underlying geometry.

## 1 Introduction

Node-link representations of graphs in the 2-dimensional Euclidean plane are the most typically used graphs visualizations. The structure of many graphs, notably planar graphs, can be realized well in the plane, but others are better represented in non-Euclidean geometries. For example, 3-dimensional polytopes are well represented in spherical space, while large hierarchies such as trees can be cleanly embedded in hyperbolic space. Standard hyperbolic projections into Euclidean space also provide a natural "focus+context" view of the graph, with parts of the graph near the center of the view shown large and those far from the center progressively smaller. There is also some evidence that hyperbolic geometry often underlies complex networks [21]. Though there has been some work on visualizing hierarchies using hyperbolic space in the browser [14], there are no tools that support browser-based hyperbolic visualization of general graphs.

We describe two methods to laying out graphs in the 2-dimensional hyperbolic space, $H^2$. The first method relies on taking a pre-computed Euclidean layout of a graph and projecting it into hyperbolic space, providing standard map interactions, such as pan, zoom, re-center, click and drag. We implement this method in GMap, a graph visualization system that provides several layout algorithms for node-link and map-based visualization. This allows us to view and interact with GMaps, MapSets, BubbleSets, and LineSets in hyperbolic space. The second method makes use of a generalization of force-directed algorithms to Riemannian geometries [20]. We exploit the locally Euclidean properties of hyperbolic space so that together with Möbius transformations we can accurately

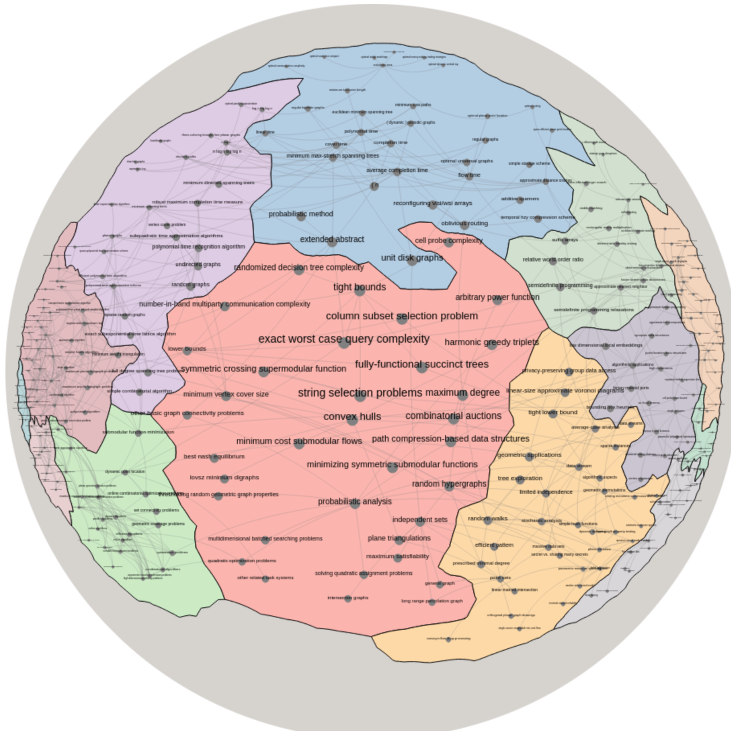



**Fig. 1.** Hyperbolic realization of the SODA graph described in Appendix A using a GMap style visualization.

model the forces. In particular, this approach allows us to compute layouts where distances between nodes in hyperbolic space correspond to the underlying graph-theoretic distances between them.

## 2 Related Work

The graph layout problem typically involves placing nodes and routing edges in 2-dimensional Euclidean space. Force-directed algorithms model the system as a set of springs and attempt to balance the forces on nodes. Both their conceptual simplicity and their generally aesthetically pleasing results have made this class of algorithms particularly useful for computing graph layouts [19]. Force-directed algorithms have been generalized to Riemannian geometries, (e.g., spherical and hyperbolic) by computing tangent planes at each node [20].

To the best of our knowledge, there are no browser-based tools for visualizing general graphs in hyperbolic space. One of the earliest approaches by Lamping *et al.* [23] embeds hierarchies into the hyperbolic plane by recursively placing each node's children evenly spaced around the arc of a circle. This is possible thanks to the exponential expansion intrinsic to the geometry. They make use of the Poincarè projection to display the graph on the computer monitor, which also provides the now well known "focus+context" effect. Navigating the hierarchy is done by re-centering the projection at a new point in the hyperbolic plane. The embedding can be computed in linear time and arbitrary graphs can also be visualized using this approach by utilizing a spanning tree of the graph and "filling in" the rest of the edges later.

A bioinformatics-motivated java application by Bingham and Sudarsanam [6] uses a similar approach to visualize phylogenetic trees. Andrews *et al.* [2] also rely on Lamping *et al.*'s work in their Hierarchy Visualization System as do Baumgartner and Waugh [4] who visualize Roget's thesaurus. The Java InfoVis Toolkit also implements a hyperbolic hierarchy browser [5] and TreeBolic implements the hyperbolic tree layout [8]. More recently, Glatzhofer developed a hyperbolic hierarchy browser utilizing d3.js, a javascript graphics library which works in the browser, and can display large hierarchies smoothly with several different layout algorithms [13,14,15].

While most prior work considers the 2-dimensional hyperbolic plane, Munzner has also used 3D hyperbolic space to visualize hierarchies with the help of the Beltrami-Klein projection [26,27,28,29]. Here geodesics are mapped to straight lines rather than the circular arcs of the Poincarè projection. Munzner's work has been re-implemented in two subsequent systems: Walrus [17] and h3py [39].

Hyperbolic space has been explored in the context of dimensionality reduction, specifically multi-dimensional scaling (MDS). This technique attempts to closely match pairwise similarities with distances in an embedding; the more similar two elements are, the closer they are in the embedding. The Euclidean distance is traditionally used as a closeness metric but Walter and Ritter describe a method that replaces the Euclidean distance metric with hyperbolic distance [38]. Computing a graph layout can be interpreted as an MDS problem by treating the graph theoretic distance between pairs of nodes as their pairwise similarity metric. It has been shown that some graphs can achieve lower distortion with fewer dimensions in hyperbolic space than in Euclidean space [7].

Recently, an open-source hyperbolic visualization tool RogueViz includes many projections and educational tools, although its restriction to tessellations of the hyperbolic plane make it less than ideal for graph drawing [9]. Self-organizing maps have been generalized to hyperbolic space, but are restricted to lattices [31].

## 3 Projection-based Method

The first method we present is based on the idea of starting with a precomputed layout and projecting it to the hyperbolic plane. The implementation is available at `http://gmap.cs.arizona.edu` under "advanced options." GMap is a browser based graph visualization system that offers several layout algorithms, clustering algorithms and visualization styles, focused on map-like representations [12]. In addition to node-link diagrams, GMap provides different style visualizations, in-

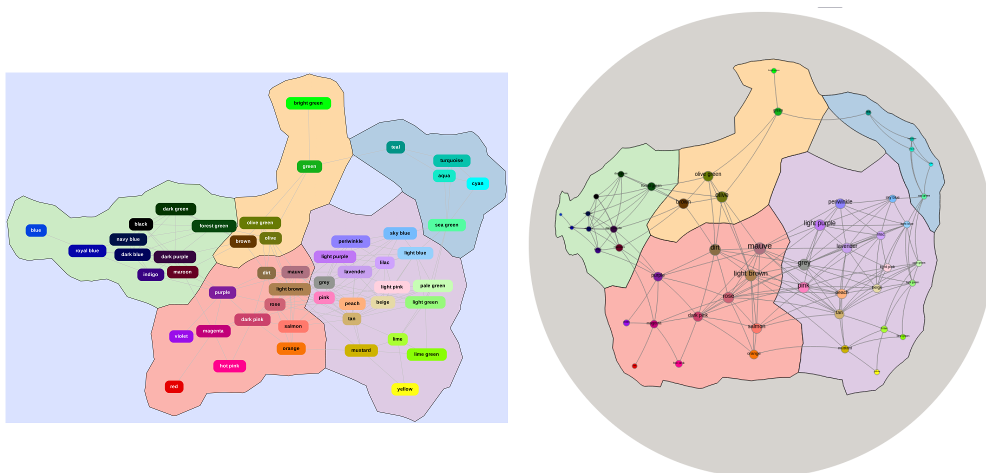



**Fig. 2.** On the left, a Euclidean map of a GMap graph. On the right, its Hyperbolic inverse projection.

cluding MapSets [11], BubbleSets [10], and LineSets [1]. Several human-subject studies suggest that such map-like visualizations are at least as good as traditional node-link diagrams when it comes to task performance, memorization, and recall of the data [33,34].

GMap layouts are saved in the graphviz DOT file format which includes graph-wide attributes, a node list, and an adjacency list. Additionally, each node and edge have a set of attributes. GMap computes the layout and stores their position as Cartesian coordinates. This is sufficient to draw node-link diagrams and the other map-like layouts: GMaps, MapSets, BubbleSets and LineSets. For these, polygons given as a set of vertices are stored as a graph-wide attribute along with colors. The parsing of these polygons is achieved using the same implementation as Perry *et al.* [32].

GMap relies on two different layout algorithms for computing a Euclidean layout: *sdfp* is a multi-level force-directed algorithm [16] and *neato* is a implementation of the Kamada-Kawai algorithm [18]. Figure 4 shows an example of the *colors graph* drawn as a node-link and GMap diagram. This is a graph of the 38 most popular RGB colors, courtesy of xkcd[1]. Figure 3 shows the same graph using the BubbleSets and LineSets options.

We make use of a javascript library called Hyperbolic Canvas [3]. It is a mathematical model of the Poincarè disk projection of hyperbolic space that allows lines and shapes to be drawn using an HTML canvas. The projection-based pipeline below is based on the approach by Perry *et al.* for browser-based visualization of graphs on the sphere [32].

[1] https://xkcd.com/color/rgb/

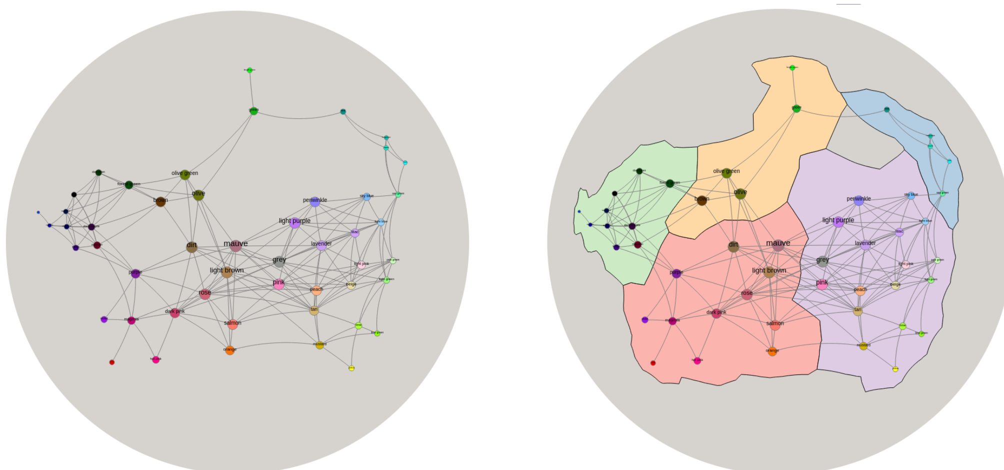



**Fig. 3.** A node-link and GMap diagram of the colors graph.

### 3.1 The Projection-based Pipeline

Given a pre-computed 2-dimensional Euclidean layout, the projection-based method can be summarized as follows:

1. Calculate geometric mean of the 2-d Euclidean layout
2. Apply an inverse hyperbolic Lambert azimuthal projection centered on the geometric mean
3. Project back into the Euclidean plane of the browser using the Poincarè projection (providing the look and feel of hyperbolic space).

**Hyperbolic projections:** It is well known that non-Euclidean spaces (such as spherical and hyperbolic spaces) can not be perfectly projected to the Euclidean plane. No matter what type of projection is used, something will get lost in the translation: distances are distorted, or region areas are distorted, or angles are distorted. This problem is well studied in cartography in the context of projecting the sphere onto the 2-d Euclidean plane.

Knowing that a perfect embedding in the plane is impossible, useful maps can still be created by choosing which information to preserve. Three well-known projections of the sphere are gnomonic, orthographic, and stereographic projections. The gnomonic projection preserves straight lines; geodesics of the sphere are shown as straight lines in the projection. This is particularly useful in flight planning, and is said to be the oldest map projection. The orthographic projection resembles the view of the Earth from space, and preserves scale at the center of the projection, making it useful in visualization. Finally, the stereographic projection preserves angles and has its roots in star charts used in sailing [37]. Of course, many more spherical projections exist, inspiring this xkcd comic.[2]

[2] https://xkcd.com/977/

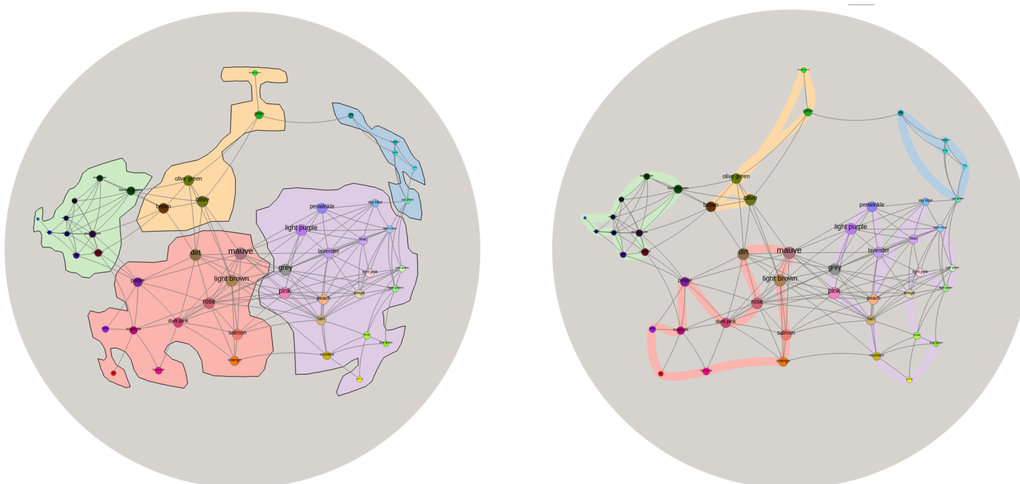



**Fig. 4.** BubbleSets and LineSets visualizations of the colors graph.

Hyperbolic surface is curved (negatively) just like spherical space is curved (positively), resulting in similar problems when attempting to display it in the plane of a monitor or on a piece of paper. Just as the sphere has many projections that serve different purposes, so there exists many hyperbolic projections to the plane, although they are not as well studied. These projections can be thought of as analogous to their spherical counterparts. Not only do they preserve the same information but they can often be derived in an analogous way. For instance, the Beltrami-Klein projection is analogous to the gnomonic projection of the sphere; they both preserve geodesics as straight lines. Similarly, the Gans model of the hyperbolic plane is analogous to the orthographic projection, in that they both have a point of perspective at infinity. The Poincarè projection is a spherical analogue of the stereographic projection as they both preserve angles.

**Hyperbolic Lambert azimuthal projection** Since we know we are projecting node-link and map diagrams, it seems reasonable to choose to preserve areas. One way this can be accomplished is through a less common hyperbolic analogue to the Lambert azimuthal projection, which has been called the hyperbolic Lambert azimuthal projection. This projection is equi-areal, so area is preserved. The hyperbolic analogue can be derived in much the same way as the sphere.

Consider two discs: one in the 2-dimensional Euclidean space and the other in the 2-dimensional hyperbolic space. Denote the area of the Euclidean disk of radius $r$ as $e(r)$ and the area of a hyperbolic disk of the same radius $h(r)$. We want a function $f(r)$ such that $e(r) = h(f(r))$. Assuming unit curvature, then

$$h(r) = 2\pi(cosh(r) - 1)$$

$$e(r) = \pi r^2$$

Then substitution reveals

$$f(r) = arccosh(\frac{1}{2}r^2 + 1),$$

where *arccosh* is the inverse hyperbolic cosine. Note that this is the inverse projection as we go from the Euclidean plane to the hyperbolic plane. This gives us the transformation $(r, \theta) \rightarrow (f(r), \theta)$, which preserves areas, but distorts angles and shapes. The further away a shape is from the projection center the greater the distortion, so centering about the geometric mean reduces this effect.

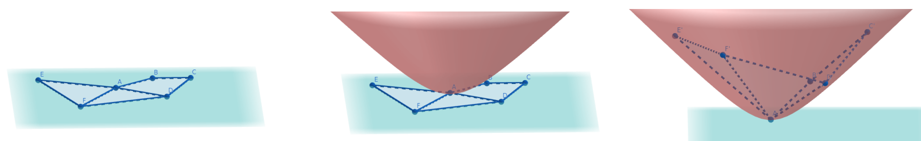


**Fig. 5.** An illustration of the inverse hyperbolic Lambert azimuthal projection from the 2-d plane to the surface of a hyperboloid.

**Poincarè projection** Recall that the Poincarè projection of the hyperbolic plane is similar to the stereographic projection of the sphere, in that it preserves angles. The infinite hyperbolic plane is mapped to the inside of the unit disk with hyperbolic lines corresponding to either arcs of circles orthogonal to the boundary of the disk, or diameters of the disk if the line passes through the origin. The Poincarè disk intrinsically provides the look and feel of hyperbolic space in the browser. The "focus+context" mentioned before is due to the Poincarè projection. A small area near the border of the disk represents a very large area in hyperbolic space, while the same size area near the center of the disk represents a small area of hyperbolic space. This can be seen mathematically in the transformation that takes the hyperbolic plane to the Poincarè disk

$$(r, \theta) - > (\frac{e^r - 1}{e^r + 1}, \theta)$$

The exponentiation in the Poincarè transformation implies a practical limit on the hyperbolic radius of about 700, as larger values require dealing with very large numbers and lead to numerical overflow.

### 3.2 Interactive Features

**Navigating the Map:** One of the main reasons for using map-like visualization for graphs is our familiarity with map interactions such as pan, zoom, click and drag. In the Poincarè disk, clicking and dragging brings new nodes and regions into focus, allowing the viewer to exploit the "focus+context" property of the projection. We accomplish this by making use of Möbius transformations.

A Möbius transformation is a complex function of the form $f(z) = \frac{az+b}{cz+d}$ where $z$ is a complex variable and $ad - bc \neq 0$. Möbius transformations have many uses in complex analysis and geometry, but one subgroup is especially useful for our purposes; the class of transformations that map the open unit disk to itself. In particular the transformation

$$f(z) = \frac{z - z_0}{-\tilde{z_0} z + 1}$$

takes $z_0$ to the origin and preserves the Poincarè projection of the hyperbolic plane, i.e., the transformation recenters the Poincarè projection at $z_0$.

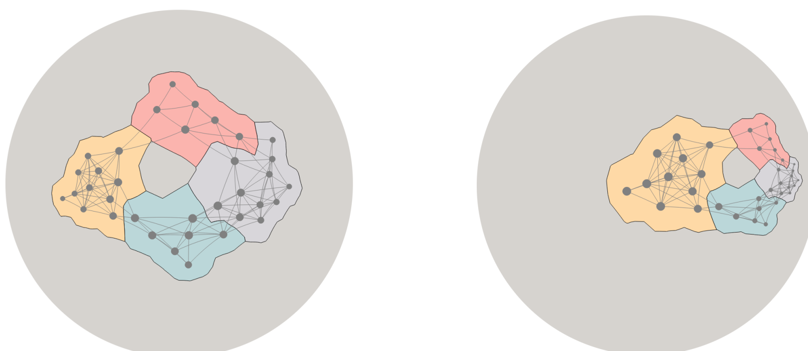


**Fig. 6.** The same graph centered about two different origins.

We can obtain transitions that look smooth to the human eye by repeatedly applying the above transformation at a point some $\epsilon$ distance from the previous origin in the direction the mouse is being dragged. Two still images centered at different points in a random graph are shown in Figure 6, but interacting with the actual visualization in GMap best conveys the idea.

**Zoom and Coverage:** We define two different notions of scale on the Poincaré disk that a viewer can adjust via sliders. The notion of zooming is fairly striaghtforward and controls the size of the Poincarè disk in the browser window. Intuitively, the zoom slider brings the disk closer or further away from the point of perspective; see Fig 7. Coverage controls the total area the layout occupies, which poses interesting challenges. As a consequence of Euclid's parallel postulate not holding in hyperbolic space, the hyperbolic plane is not invariant to scale [35]. However, we can re-scale the layout while it is still in the Euclidean plane, before we project it to $H^2$. We do just that and by default we use a scaling factor of 0.005, which works well for the layouts that GMap generates. The coverage slider allows us to change the size of the layout in the Poincarè disk.

Lamping *et al.* [23] proposed to adjust the radial component using a nonconformal mapping that increases the distance an object was from the center

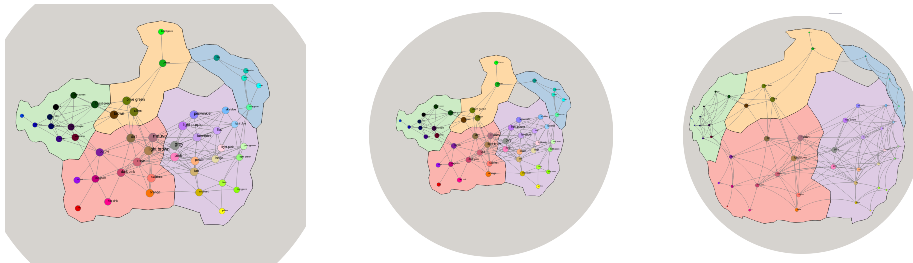


**Fig. 7.** An example of the default layout (center), increased zoom (left), and increased coverage (right).

of the disk. This method is not well-suited for our purposes, however, as it increases detail near the center of the layout, but distorts shapes further away. We instead exploit the friendly properties of the Euclidean plane, by projecting the layout back to the Euclidean plane (using the hyperbolic Lambert azimuthal projection), applying the desired scale factor, and then re-projecting it (using the inverse of the same projection).

**Reset, Recenter, Labels and Opacity:** While a viewer need not be an expert in hyperbolic geometry to use the visualization we provide, some properties of the geometry call for additional features to make the system more useful.

Given the infinite space and "focus+context" nature of the Poincarè projection, it is not unreasonable that a viewer may lose sight of the layout or even become unsure of "where" they are. To alleviate this potential problem, we provide a *reset button*, that brings us back to the original view. A viewer can also double-click on a node/region to *recenter* the projection at that point.

The number of edges near the border in large node-link diagrams and maps might increase to a level where the layout is difficult to discern. An edge *opacity slider* helps mitigate this problem. A related problem is that labels near the border can start to overlap. Although label size scales with distance from the origin, the two scales are necessarily different (as distances can approach infinity and label sizes cannot). To help remedy this, we provide a *label size slider*, which adjusts the relative label sizes on the fly.

### 3.3 Considerations

The inherent distortion of shapes and angles introduced when using the inverse Lambert projection to the hyperbolic plane implies that at some threshold the outer regions of the layout will become too distorted to be of use. This is already apparent in the MusicLand example from GMap as shown in Figure 8, with around 250 nodes. Even though our method can handle larger graphs, it is clear that larger graphs pose additional challenges. A multi-level representation of the graph might be useful to provide "semantic zooming" where we start with a high level overview of the graph and zooming in brings up more details, following Schneiderman's mantra (overview first, zoom and filter, details on demand). Different inverse hyperbolic projections might also be useful and worth exploring.

When moving through a curved space, an inherent property causes an observer to incur rotation. This could be desirable, as it gives several different perspectives on the same layout, but it could potentially be confusing when navigating large maps. Specifically, moving the layout in the Poincarè disk, incurs a rotation in the layout (clockwise or counter-clockwise): consider translating a layout some fixed distance up, the same distance to the right, then again down, and back to the left. In 2D Euclidean geometry, the layout would be identical after these transformations. However, in the Poincarè disk (and hyperbolic geometry in general) this will cause the layout to be rotated about 90 degrees from its original orientation. An orientation correcting transformation could be applied after translating the layout, but in our prototype we only provide the "reset button," which resets the original layout.

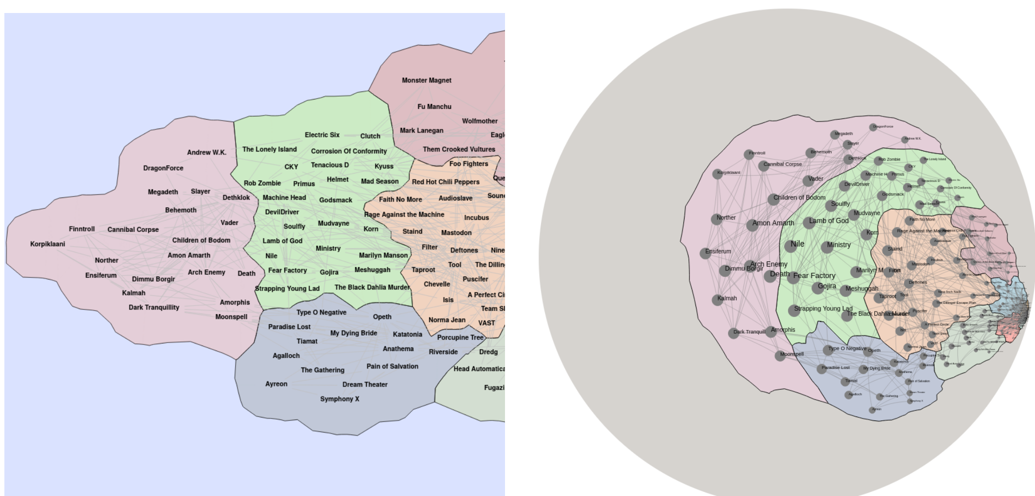



**Fig. 8.** A 2D Euclidean GMap instance of the MusicLand graph (left) and its hyperbolic realization (right).

## 4 Force-directed Method

Our projection-based hyperbolic visualization method uses a precomputed 2-dimensional Euclidean layout, but it uses hyperbolic space just for the visualization and "focus+context" effect, rather than for the actual graph embedding. Properly embedding the graph in hyperbolic space would allow us to take advantage of the underlying hyperbolic geometry. Algorithms for directly embedding special classes of graphs in hyperbolic space, such as trees and hierarchies, can better take advantage of the properties of the space and obtain better embeddings than via projections. It is also possible to modify the standard force-directed algorithm for operation in Riemannian geometries (such as hyperbolic and spherical) by taking advantage of the locally Euclidean properties of such spaces [20]. The implementation, which provides visualization in the browser, can be found at `https://github.com/Mickey253/hyperbolic-space-graphs`.

The idea is to compute a tangent plane at each vertex embedded in the non-Euclidean Riemannian space, mapping every other vertex to that plane, performing a step of a force-directed algorithm in the plane, and projecting back the resulting node position changes to the Riemannian space. While conceptually simple, this method allows the graph to make use of the unique properties of the corresponding non-Euclidean geometry. For instance, on the sphere, 3-dimensional polytopes wrap "around" the sphere, more accurately realizing their structure than anything that can be done in the plane [32].

We apply this idea to the Kamada-Kawai type of force-directed graph layout algorithm, for its conceptual simplicity and its desirable property of capturing graph structure (e.g., graph distances between pairs of nodes) in the embedding (e.g., realized distances between pairs of nodes in the non-Euclidean space). Specifically, we compute the graph theoretic distances between all pairs of nodes and these define desired distances in the layout. Spring forces, proportional to the squared Euclidean distance between nodes in the layout, are used to gradually improve a given initial layout to one in which realized distances match the graph theoretic distances [18]. Formally, there is an attractive or repulsive force (similar to stress) defined for any pair of edges based on the difference between the graph theoretical distance and the realized distance in the current embedding. Specifically, the total energy of the system is modeled as:

$$E = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{2} k_{ij} (|p_i - p_j| - d_{ij})^2$$

where given a pair of nodes $i$ and $j$ $d_{ij}$ is the graph theoretic distance between them, $|p_i - p_j|$ is the current realized distance in the embedding between them, and $k_{ij}$ is the strength of the spring forces between them. The layout is obtained by reducing the energy of the system via gradient descent.

### 4.1 Tangent plane

In order to compute a tangent plane at some point $x$ in $H^2$, we need to set the distance between $x$ and every other point in the plane to the hyperbolic distance between them, and ensure the angle between the points stays the same [20]. The Poincarè disk preserves angles, so we only need to map hyperbolic to Euclidean distances. In the Poincarè model, hyperbolic distance is simplest to compute when one point is at the origin, so we first apply the Möbius transformation

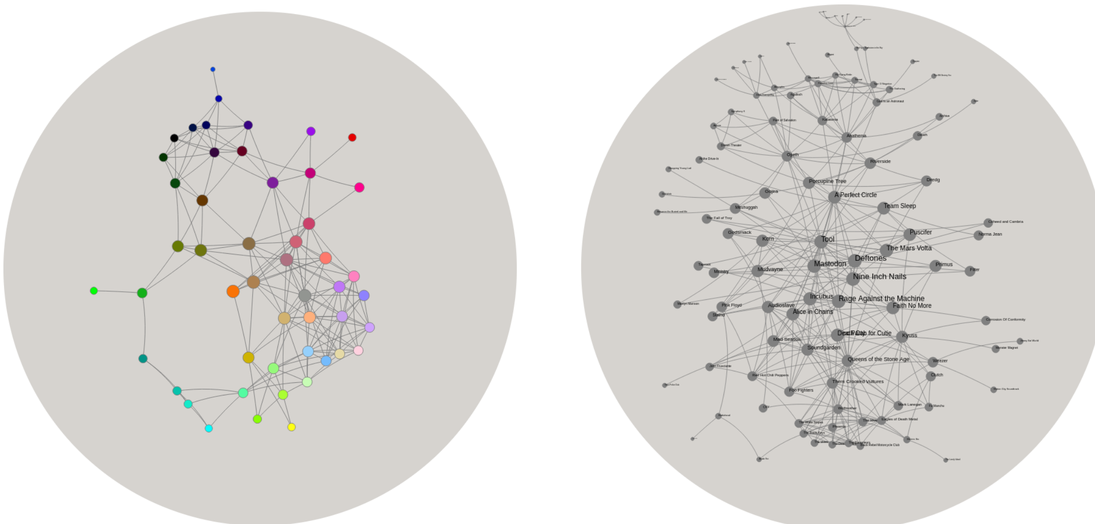



**Fig. 9.** Force-directed colors (left) and MusicLand (right) graphs.

that takes $x$ to the origin. The distance between $x$ and any point $y$ is

$$d_h(x,y) = ln(\frac{1+|y|}{1-|y|}) = 2arctanh\,|y|$$

where arctanh is the inverse hyperbolic tangent. Then, let $x$ be the center of the disk and for every point $y$, let the transformation $(|y|, \theta) \rightarrow (d_h(x,y), \theta)$ be its location in the tangent Euclidean plane, using polar coordinates.

Once the tangent plane is computed and a step of the force-directed algorithm has completed, the central node must be placed back into hyperbolic space. This is accomplished through an inverse of the above equations. Let $y'$ be the new location of the moved node. We apply the transformation

$$(|y'|, \theta) -> (tanh\,\frac{|y'|}{2}, \theta)$$

The following Möbius transformation takes the disk back to its former origin, $z_0$

$$f^{-1}(y) = \frac{-y - z_o}{-\tilde{z_o}y - 1}$$

### 4.2 Precision

While hyperbolic geometry poses many interesting challenges for graph drawing, the most notable we encountered was the issue of precision. It is well known that floating point numbers are not arbitrarily precise and that this can cause problems when the number of significant bits needed is large. This effect is pronounced on the Poincarè disk, as the number of bits needed to accurately reflect the hyperbolic position increases exponentially as one approaches the border of the disk. We choose to trade accuracy for stability: if a node is pushed to a within 0.001 of the border, then the node is "pulled back" so it does not disappear altogether. This effectively creates a bounding region defined by a circle of (Euclidean) radius 0.999 from the center of the Poincarè disk.

### 4.3 Considerations

Unlike the sphere where there are natural graphs that can be realized on the sphere but not the plane (e.g., 3-dimensional polytopes), it is not clear what are simple examples that illustrate that our method is in fact making good use of the hyperbolic geometry. A path does converge to a geodesic in our implementation, so this is a good indication. Trees span out evenly as one might expect in hyperbolic space, and several other layouts are shown in Appendix A.

Currently our hyperbolic force directed approach only works for node-link diagrams. Extending its utility to map-like visualizations (such as GMap, BubbleSets, MapSets, LineSets) could be accomplished by projecting the completed layout of the graph onto the plane, computing the needed clusters and polygons, then projecting them back onto the Poincarè disk. Better yet, we could perform the clustering and polygon-generation directly in hyperbolic space. The cluster regions (polygons) are computed using Voronoi diagrams in GMap and it has been shown that Voronoi diagrams for 2-dimensional points generalize to hyperbolic space and can be computed in $O(n \log n)$ time [30].

## 5 Discussion, Conclusions and Future Work

We described two methods for visualizing graphs in hyperbolic space. The projection-based method is fully functional at http://gmap.cs.arizona.edu and provides all the layout algorithms, clustering algorithms and visualization styles available within GMap. The projection-based method allows us to show any 2-dimensional Euclidean visualization in hyperbolic space, where we can take advantage of the "focus+context" properties of the space while still relying on standard map interactions. The method currently relies on Lambert azimuthal projections and the Poincarè disk model. We have not yet explored other projections or the Beltrami-Klein model. Finally, this method does not fully take advantage of the underlying geometry of the space.

The force-directed method uses the non-Euclidean geometry better and is also fully functional at https://github.com/Mickey253/hyperbolic-space-graphs. This method does utilize the geometry of hyperbolic space, but is not as efficient as the projection-based one. The underlying Kamada-Kawai algorithm is already rather computationally expensive with a pre-processing of an all-pairs shortest path (implemented with an $O(|V|^3)$ algorithm, where $|V|$ is the number of nodes in the graph) and the many tangent plane computations are also time-consuming. It can take nearly one minute to compute a layout for graph with 100 nodes, and the running time becomes prohibitively expensive for larger graphs. We have not yet implemented scalable force-directed algorithms, or a multi-level ones, and this is a natural direction for future work.

While there is good evidence for the potential advantage of visualizing graphs in non-Euclidean spaces, we are not aware of human-subjects studies evaluating the effectiveness of such of non-Euclidean graph visualization. Now that GMap conveniently includes Euclidean layouts as well as spherical and hyperbolic realizations this may be something to consider. We have not performed quantitative evaluations either, besides measuring runtimes and there are several adjustable parameters that can and should be optimized.

We have also not yet evaluated whether graphs that are supposed to be better embeddable in hyperbolic space [7], do indeed realize their underlying structure better in hyperbolic than in Euclidean space. Virtual reality and augmented reality provide promising applications for hyperbolic visualization, although prior work seems to have only considered spherical space in this context [22]. Finally, we want to compute a graph layout directly in hyperbolic space (not relying on the Euclidean plane at all) which requires generalizations to non-Euclidean geometry of Euclidean techniques such as MDS [36], t-SNE [24] and UMAP [25].

## References

1. Alper, B., Riche, N.H., Ramos, G.A., Czerwinski, M.: Design study of LineSets, a novel set visualization technique. IEEE Trans. Vis. Comput. Graph. **17**(12), 2259–2267 (2011). https://doi.org/10.1109/TVCG.2011.186
2. Andrews, K., Putz, W., Nussbaumer, A.: The hierarchical visualisation system (HVS). In: 11th International Conference on Information Visualisation, IV 2007, 2-6 July 2007, Zürich, Switzerland. pp. 257–262. IEEE Computer Society (2007). https://doi.org/10.1109/IV.2007.112
3. Barry, N.: Hyperbolic Canvas Github Page. `https://github.com/ItsNickBarry/hyperbolic-canvas`, accessed: 2021-06-06
4. Baumgartner, J., Waugh, T.A.: Roget2000: a 2d hyperbolic tree visualization of Roget's thesaurus. In: Erbacher, R.F., Chen, P.C., Gröhn, M., Roberts, J.C., Wittenbrink, C.M. (eds.) Visualization and Data Analysis 2002, San Jose, CA, USA, January 19, 2002. SPIE Proceedings, vol. 4665, pp. 339–346. SPIE (2002). https://doi.org/10.1117/12.458803
5. Belmonte, N.G.: Javascript InfoVis Toolkit. `https://philogb.github.io/jit/demos.html`, accessed: 2021-06-06
6. Bingham, J., Sudarsanam, S.: Visualizing large hierarchical clusters in hyperbolic space. Bioinform. **16**(7), 660–661 (2000). https://doi.org/10.1093/bioinformatics/16.7.660
7. Bläsius, T., Friedrich, T., Krohmer, A., Laue, S.: Efficient embedding of scale-free graphs in the hyperbolic plane. IEEE/ACM Trans. Netw. **26**(2), 920–933 (2018). https://doi.org/10.1109/TNET.2018.2810186
8. Bou, B.: Treebolic2 Webpage. `http://treebolic.sourceforge.net/treebolic2/en/index.html`, accessed: 2021-06-06
9. Celinska, D., Kopczynski, E.: Programming languages in GitHub: A visualization in hyperbolic plane. In: Proceedings of the Eleventh International Conference on Web

and Social Media, ICWSM 2017, Montréal, Québec, Canada, May 15-18, 2017. pp. 727–728. AAAI Press (2017), https://aaai.org/ocs/index.php/ICWSM/ICWSM17/paper/view/15583
10. Collins, C., Penn, G., Carpendale, S.: BubbleSets: Revealing set relations with isocontours over existing visualizations. IEEE Trans. Vis. Comput. Graph. **15**(6), 1009–1016 (2009). https://doi.org/10.1109/TVCG.2009.122
11. Efrat, A., Hu, Y., Kobourov, S., Pupyrev, S.: MapSets: Visualizing embedded and clustered graphs. J. Graph Algorithms Appl. **19**(2), 571–593 (2015). https://doi.org/10.7155/jgaa.00364
12. Gansner, E.R., Hu, Y., Kobourov, S.: GMap: Visualizing graphs and clusters as maps. In: IEEE Pacific Visualization Symposium PacificVis 2010, Taipei, Taiwan, March 2-5, 2010. pp. 201–208. IEEE Computer Society (2010). https://doi.org/10.1109/PACIFICVIS.2010.5429590
13. Glatzhofer, M.: Hyperbolic tree of life. https://hyperbolic-tree-of-life.github.io/, accessed: 2021-06-06
14. Glatzhofer, M.: Hyperbolic Browsing. Master's thesis, Institute of Interactive Systems and Data Science (ISDS),Graz University of Technology, Austria (2018)
15. Glatzofer, M.: d3-hypertree Github page. https://github.com/glouwa/d3-hypertree, accessed: 2021-06-06
16. Hu, Y.: Efficient, high-quality force-directed graph drawing. Mathematica journal **10**(1), 37–71 (2005)
17. Hyun, Y.: https://www.caida.org/catalog/software/walrus/#H2540 (2000), accessed: 2021-06-06
18. Kamada, T., Kawai, S.: An algorithm for drawing general undirected graphs. Inf. Process. Lett. **31**(1), 7–15 (1989). https://doi.org/10.1016/0020-0190(89)90102-6
19. Kobourov, S.: Spring embedders and force directed graph drawing algorithms. CoRR **abs/1201.3011** (2012), http://arxiv.org/abs/1201.3011
20. Kobourov, S., Wampler, K.: Non-Euclidean spring embedders. IEEE Trans. Vis. Comput. Graph. **11**(6), 757–767 (2005). https://doi.org/10.1109/TVCG.2005.103
21. Krioukov, D.V., Papadopoulos, F., Kitsak, M., Vahdat, A., Boguñá, M.: Hyperbolic geometry of complex networks. CoRR **abs/1006.5169** (2010), http://arxiv.org/abs/1006.5169
22. Kwon, O., Muelder, C., Lee, K., Ma, K.: A study of layout, rendering, and interaction methods for immersive graph visualization. IEEE Trans. Vis. Comput. Graph. **22**(7), 1802–1815 (2016). https://doi.org/10.1109/TVCG.2016.2520921
23. Lamping, J., Rao, R., Pirolli, P.: A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In: Katz, I.R., Mack, R.L., Marks, L., Rosson, M.B., Nielsen, J. (eds.) Human Factors in Computing Systems, CHI '95 Conference Proceedings, Denver, Colorado, USA, May 7-11, 1995. pp. 401–408. ACM/Addison-Wesley (1995). https://doi.org/10.1145/223904.223956
24. Van der Maaten, L., Hinton, G.: Visualizing data using t-sne. Journal of machine learning research **9**(11) (2008)
25. McInnes, L., Healy, J.: UMAP: uniform manifold approximation and projection for dimension reduction. CoRR **abs/1802.03426** (2018), http://arxiv.org/abs/1802.03426
26. Munzner, T.: H3: laying out large directed graphs in 3d hyperbolic space. In: 1997 IEEE Symposium on Information Visualization (InfoVis '97), October 18-25, 1997, Phoenix, AZ, USA. pp. 2–10. IEEE Computer Society (1997). https://doi.org/10.1109/INFVIS.1997.636718

27. Munzner, T.: Exploring large graphs in 3d hyperbolic space. IEEE Computer Graphics and Applications **18**(4), 18–23 (1998). https://doi.org/10.1109/38.689657, https://doi.org/10.1109/38.689657
28. Munzner, T.: Interactive visualization of large graphs and networks. Ph.D. thesis, Stanford University (2000)
29. Munzner, T., Burchard, P.: Visualizing the structure of the world wide web in 3d hyperbolic space. In: Nadeau, D.R., Moreland, J.L. (eds.) Proceedings of the 1995 Symposium on Virtual Reality Modeling Language, VRML 1995, San Diego, CA, USA, December 14-15, 1995. pp. 33–38. ACM (1995). https://doi.org/10.1145/217306.217311
30. Nielsen, F., Nock, R.: Hyperbolic voronoi diagrams made easy. In: Apduhan, B.O., Gervasi, O., Iglesias, A., Taniar, D., Gavrilova, M.L. (eds.) Prodeedings of the 2010 International Conference on Computational Science and Its Applications, ICCSA 2010, Fukuoka, Japan, March 23-26, 2010. pp. 74–80. IEEE Computer Society (2010). https://doi.org/10.1109/ICCSA.2010.37
31. Ontrup, J., Ritter, H.J.: Hyperbolic self-organizing maps for semantic navigation. In: Dietterich, T.G., Becker, S., Ghahramani, Z. (eds.) Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]. pp. 1417–1424. MIT Press (2001), https://proceedings.neurips.cc/paper/2001/hash/093b60fd0557804c8ba0cbf1453da22f-Abstract.html
32. Perry, S., Yin, M.S., Gray, K., Kobourov, S.: Drawing graphs on the sphere. In: Tortora, G., Vitiello, G., Winckler, M. (eds.) AVI '20: International Conference on Advanced Visual Interfaces, Island of Ischia, Italy, September 28 - October 2, 2020. pp. 17:1–17:9. ACM (2020). https://doi.org/10.1145/3399715.3399915
33. Saket, B., Scheidegger, C., Kobourov, S., Börner, K.: Map-based visualizations increase recall accuracy of data. Comput. Graph. Forum **34**(3), 441–450 (2015). https://doi.org/10.1111/cgf.12656
34. Saket, B., Simonetto, P., Kobourov, S., Börner, K.: Node, node-link, and node-link-group diagrams: An evaluation. IEEE Trans. Vis. Comput. Graph. **20**(12), 2231–2240 (2014). https://doi.org/10.1109/TVCG.2014.2346422
35. Sala, F., Sa, C.D., Gu, A., Ré, C.: Representation tradeoffs for hyperbolic embeddings. In: Dy, J.G., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018. Proceedings of Machine Learning Research, vol. 80, pp. 4457–4466. PMLR (2018), http://proceedings.mlr.press/v80/sala18a.html
36. Shepard, R.N.: The analysis of proximities: multidimensional scaling with an unknown distance function. i. Psychometrika **27**(2), 125–140 (1962)
37. Snyder, J.P.: Map projections: A working manual. U.S. Government Printing Office (1987). https://doi.org/10.3133/pp1395
38. Walter, J.A., Ritter, H.J.: On interactive visualization of high-dimensional data using the hyperbolic plane. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada. pp. 123–132. ACM (2002). https://doi.org/10.1145/775047.775065
39. Zhang, S., Kelleher, A.: H3py Github Page. https://github.com/buzzfeed/pyh3 (2016), accessed: 2021-06-06

# 6 Appendix A: Generated Layouts

Appendix A includes further examples of layouts generated by our force-directed layout approach that did not fit into the body of the paper. The captions of each image briefly describe the graph and parameters. The source code that generated can be found at `https://github.com/Mickey253/hyperbolic-space-graphs`, along with links to videos showcasing the algorithm in action.

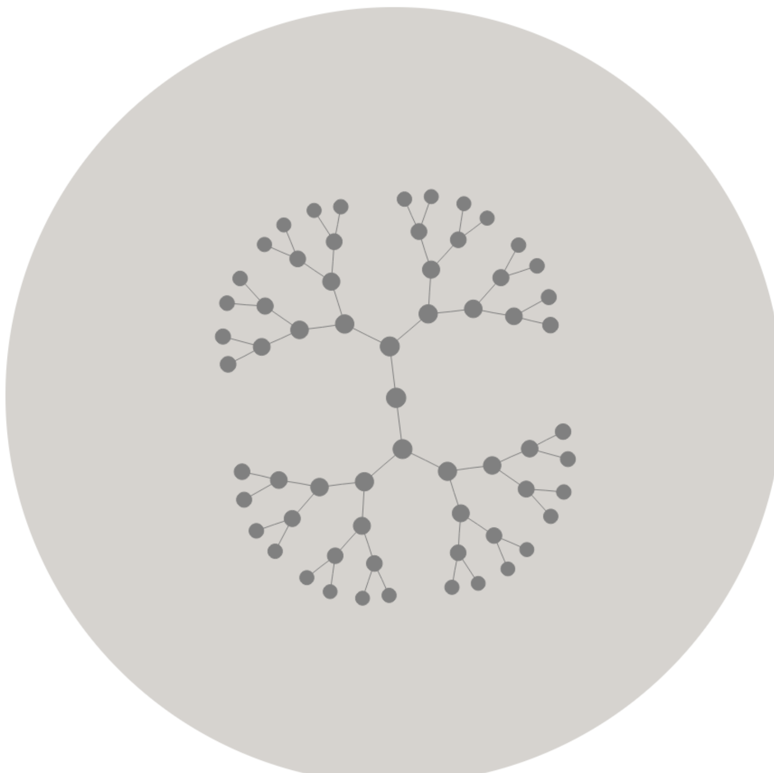
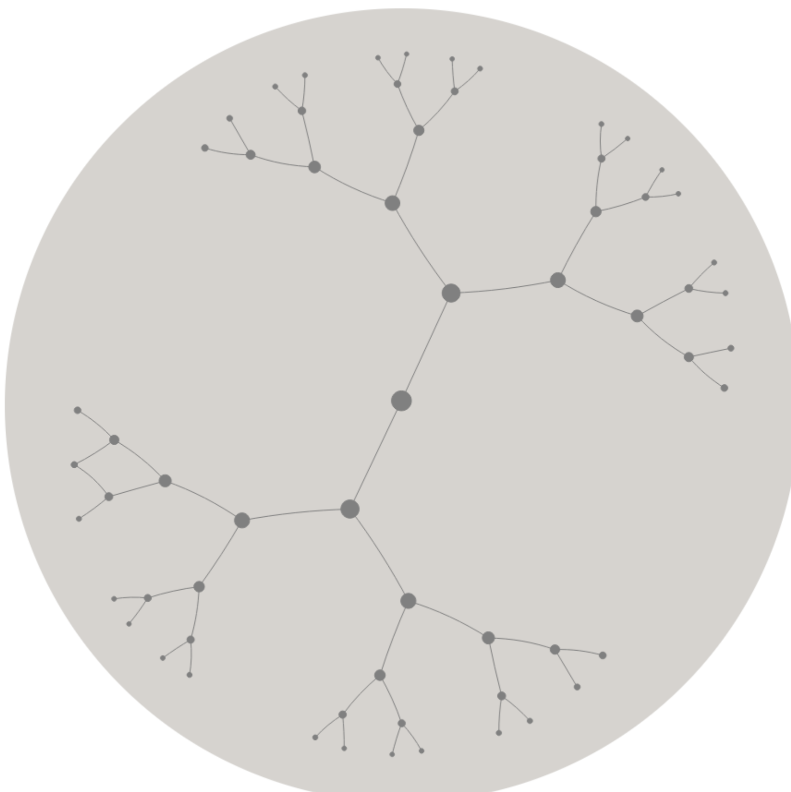


**Fig. 10.** A 63 node binary tree with edge length 2.



531 **Fig. 11.** A 63 node binary tree with edge length 5.

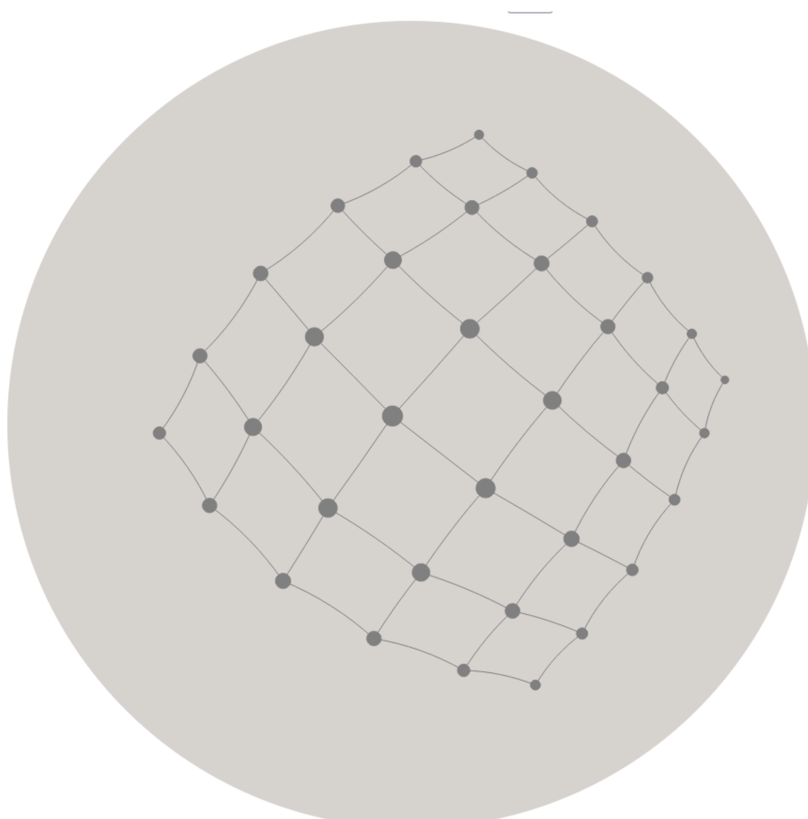


**Fig. 12.** A 6x6 grid graph with ideal edge length 5.

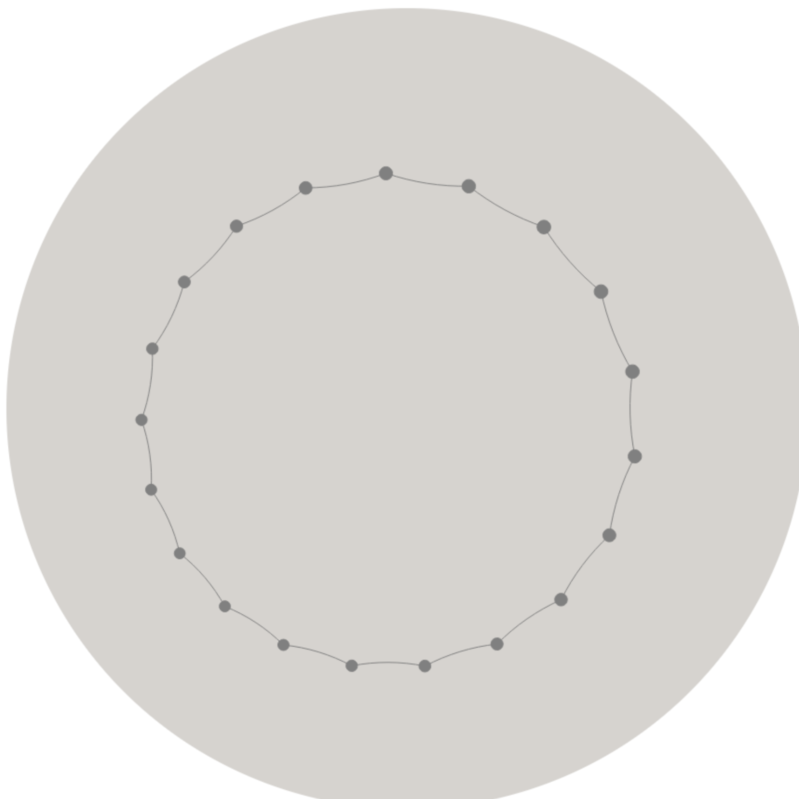


**Fig. 13.** A ring graph.

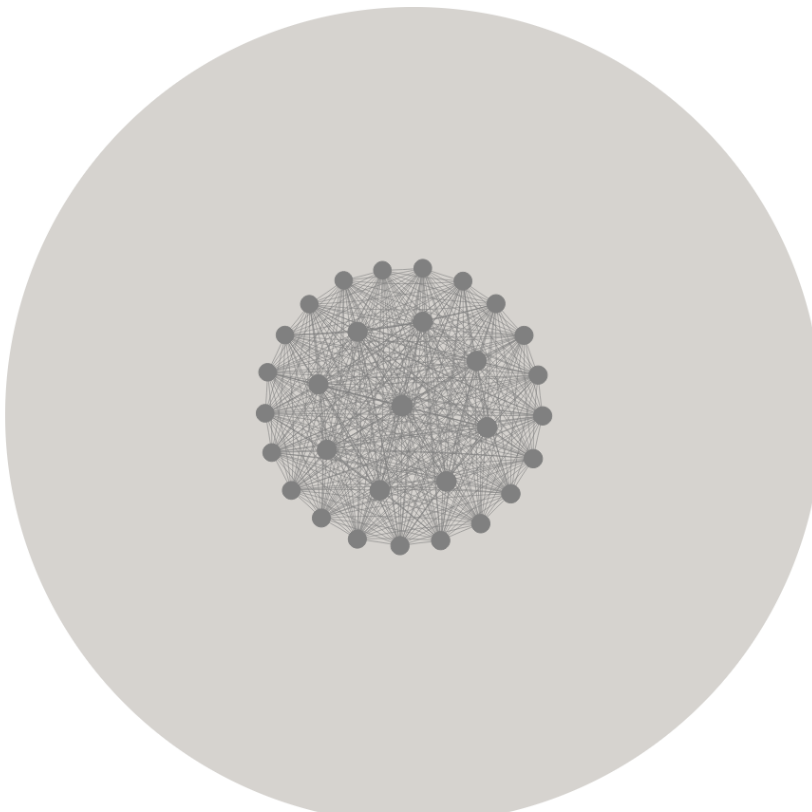


**Fig. 14.** A complete graph on 30 nodes.

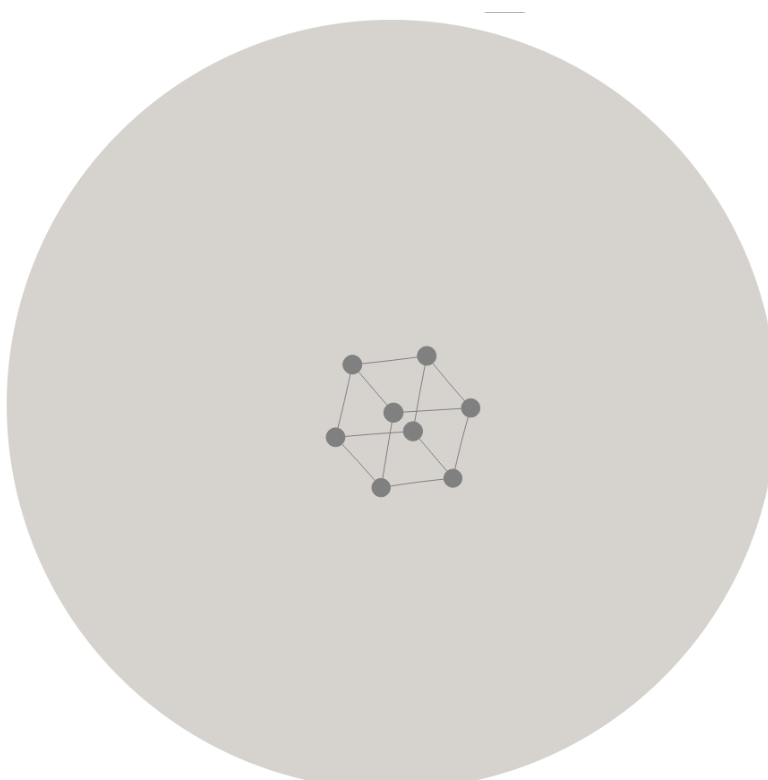
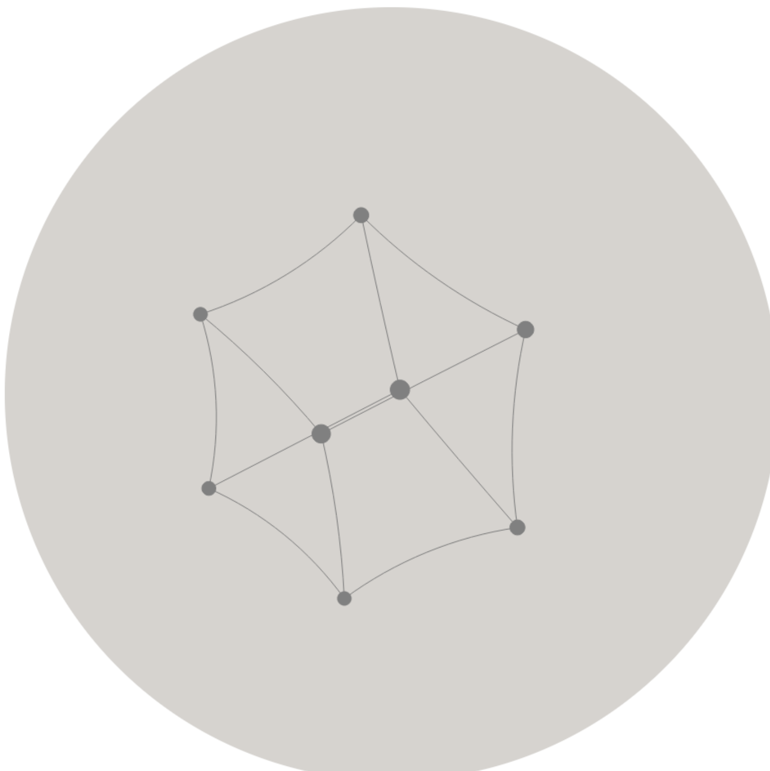
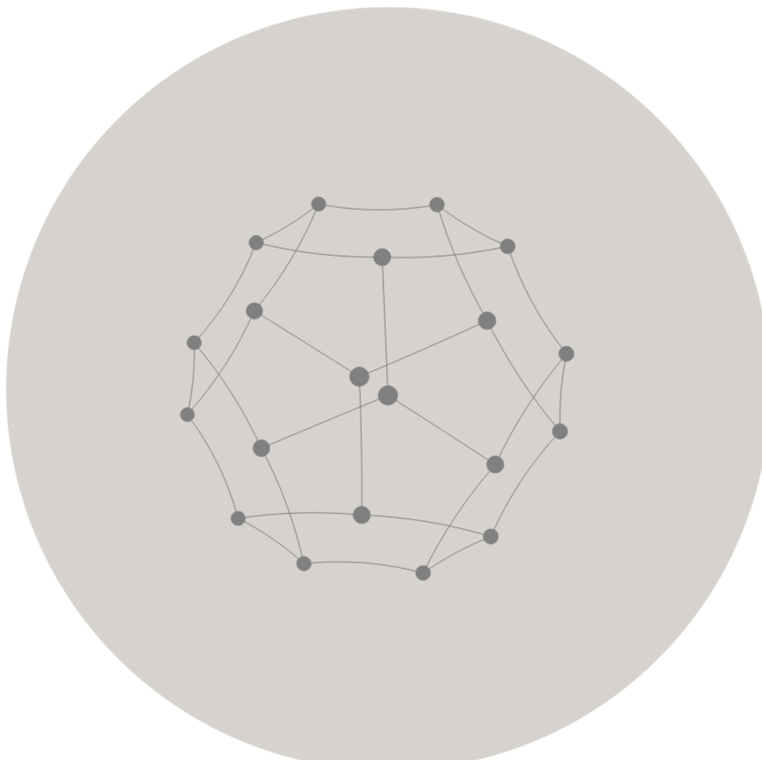


**Fig. 15.** A cube with an ideal edge length of 1.



536 **Fig. 16.** A cube with an ideal edge length of 3.



537 **Fig. 17.** A dodecahedron.