

Spherical Graph Drawing by Multi-dimensional Scaling

Jacob Miller, Vahan Huroyan, Stephen Kobourov
jacobmiller1@arizona.edu, vahanhuroyan@math.arizona.edu, and
kobourov@cs.arizona.edu

Department of Computer Science, University of Arizona

Abstract. We describe an efficient and scalable spherical graph embedding method. The method uses a generalization of the Euclidean stress function for Multi-Dimensional Scaling adapted to spherical space, where geodesic pairwise distances are employed instead of Euclidean distances. The resulting spherical stress function is optimized by means of stochastic gradient descent. Quantitative and qualitative evaluations demonstrate the scalability and effectiveness of the proposed method. We also show that some graph families can be embedded with lower distortion on the sphere, than in Euclidean and hyperbolic spaces.

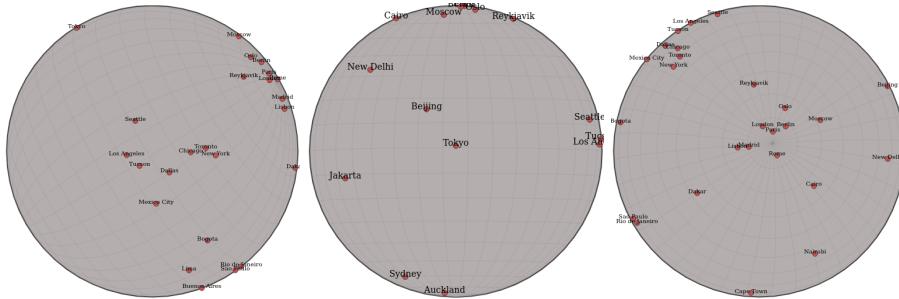


Fig. 1: Applying spherical MDS to embed 30 cities from around the Earth (given pairwise distances between the cities). The spherical MDS recovers the underlying geometry.

1 Introduction

Node-link diagrams are typically created by embedding the vertices and edges of a given graph in the Euclidean plane and different embedding spaces are rarely considered. Multi-Dimensional Scaling (MDS), realized via stress minimization or stress majorization, is one of the standard approaches to embedding graphs in Euclidean space. The idea behind MDS is to place the vertices of the graph in Euclidean space so that the distances between them are as close as possible to the given graph theoretic distances. Due to the nature of Euclidean geometry, this cannot always be done without some distortion (e.g., while K_3 naturally lives in

2D, K_4 does not, as there are no four equidistant points in the Euclidean plane). Moreover, some graphs “live” naturally in manifolds other than the Euclidean plane. For example 3-dimensional polytopes, or triangulations of 3-dimensional objects can be better represented on the sphere, while trees and special lattices are well-suited to hyperbolic spaces.

When visualizing graphs in Euclidean space, common techniques include adapting off-the-shelf dimensionality reduction algorithms to the graph setting. Such algorithms include the Multi-Dimensional Scaling (MDS) [39], Principal Component Analysis (PCA) [11], t-distributed Stochastic Neighbor Embedding (t-SNE) [26], and Uniform Manifold Approximation Projection (UMAP) [27]. The popularity of graph visualisation, and the fact that some of the underlying datasets are easier to embed in non-Euclidean spaces, led to some visualization techniques for spherical geometry [9,34] and hyperbolic geometry [20,28,37]. Most of the existing non-Euclidean graph visualization approaches, however, either lack in accuracy or do not scale to larger graphs.

With this in mind, we propose and analyze a stochastic gradient descent algorithm for spherical MDS. Specifically, we present a scalable technique to compute graph layout directly on the sphere, adapting previous work for general datasets [9] and applying stochastic gradient descent [35,42]. We provide an evaluation of the technique by comparing its speed and faithfulness to the exact gradient descent approach. We also investigate differences in graph layouts between the consistent geometries (Euclidean, spherical, hyperbolic) by first showing that *dilation or resizing* has a large effect on layouts in spherical and hyperbolic geometry, and second by showing some structures can be better represented in one geometry than the other two. All sourcecode, datasets and experiments, as well a web based visualization tool are available on GitHub: <https://github.com/Mickey253/spherical-mds>.

Note that the proposed method is not restricted to graphs, but is applicable to any dataset specifying a set of objects and pairwise distances between them.

2 Background and Related Work

We review related work in non-Euclidean geometry and graph layout methods.

2.1 Multi-dimensional Scaling

Using graph-theoretic distances to determine a graph layout dates back to the Kamada-Kawai algorithm [17]. A more general embedding approach from a given set of distances is the multi-dimensional scaling (MDS) [39] which has extensively been applied to graph layout; see [12,14,42]. Both the Kamada-Kawai and (metric) MDS algorithms aim to minimize the *stress* function, which is the sum of residual squares between the given and the embedded distances of each pair of datapoints. Formally, given a graph $G = (V, E)$ with the graph theoretic distances between its n vertices $(d_{ij})_{i,j=1}^{n,n}$, where the vertices are labeled $1, 2, \dots, n$ MDS aims to

embed the graph in \mathbb{R}^d by minimizing the following *stress* function to find the locations for its vertices:

$$\sigma(X) = \sum_{i < j} w_{ij} (||X_i - X_j|| - d_{ij})^2. \quad (1)$$

The resulting solution of $X_1, X_2, \dots, X_n \in \mathbb{R}^d$ represents the coordinates of the embedded graph vertices.

Various forms of MDS have been analyzed. Metric MDS was first studied by Shepard [39] (see equation in (3)), and the related non-metric MDS by Kruskal [22]. Classical MDS is similar but uses an objective function called *strain*.

The classical MDS has a closed form solution while the metric MDS and non-metric MDS rely on solving an optimization problems to minimize the corresponding stress functions. Many approaches have been proposed to solve (metric) MDS including stress majorization [13] and (stochastic) gradient descent [2].

When used for the purposes of visualization, the embedding space for MDS is almost always 2 dimensional Euclidean, as that is the space of a flat sheet of paper, or the flat screen of a computer monitor. The natural measure of distance is then the Euclidean norm.

In this work we will focus on metric MDS, defined in (1) but instead of embedding the graph in Euclidean space, we embed it directly on the sphere. The MDS approach has already been applied to embed graphs on spherical [9] and hyperbolic [28] spaces. Our contribution is to solve the proposed optimization problem faster and be able to handle larger graphs, address the dilation/resizing problem, as well as analyze the approach on wider range of graphs and provide a working and easy to use implementation.

2.2 Non-Euclidean Geometry

Non-Euclidean geometries are a special case of Riemannian geometries, which are spaces that are locally “smooth”: one can define an inner product on the tangent space at each point. Spherical and hyperbolic non-Euclidean geometries are similar to Euclidean geometry, except for one axiom.

Euclid’s *Elements* specify five axioms/postulates upon which all true statements about geometry should be proved. The fifth axiom is significantly more involved than the first four and mathematicians attempted for centuries to prove it using only the first four. In 1892 Lobachevsky and Bolyai independently discovered and published their formulation of hyperbolic geometry by inverting an equivalent statement to Euclid’s fifth axiom, Playfair’s axiom: *In a plane, given a line and a point not on it, at most one line parallel to the given line can be drawn through the point.* Replace “at most one line” with “at least two distinct lines” to get hyperbolic geometry. Replace “at most one line” with “there does not exist a line” to arrive at spherical geometry.

Spherical geometry has benefits in the context of data visualization. In Euclidean (or hyperbolic) layouts, one is forced to choose a “center” of the embedding, intentionally or not, whereas on the sphere there is no notion of a center.

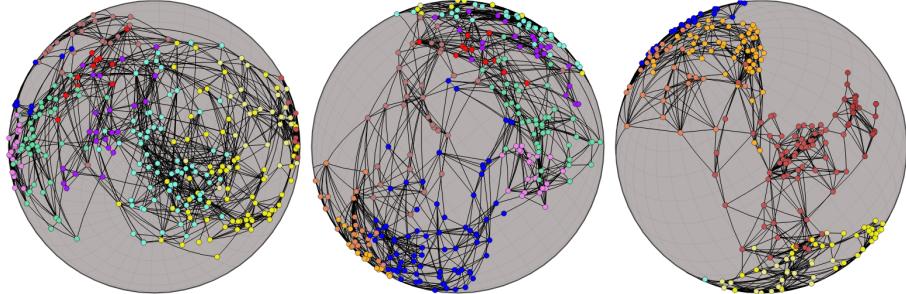


Fig. 2: The Maps of Science dataset [5] laid out using our SMDS algorithm, from three different perspectives. Each color represents a different field of science (nodes are subfields), and their relationships exhibit a ring-like structure. Any field can be placed in the center of the view.

A perceptual side effect of centered embeddings is that vertices near the center seem more important, while vertices away from the center seem more peripheral. This problem does not occur in spherical space, where simple rotation can place any vertex in the center of the view (a feature that is very useful when visualizing social networks, or networks of research fields); see Fig. 2. Additionally, many spherical projections into Euclidean space, such as the stereographic projection, provide a desirable focus+context effect.

Some focus+context type algorithms for visualizing large hierarchies by using hyperbolic geometry are discussed in [24,25].

Distances in non-Euclidean geometries generalize the concept of a straight line to that of a geodesic, defined as an arc of shortest length (not necessarily unique) that contains both endpoints. The distance between two endpoints is then the length of that curve.

A point on a sphere of radius R is uniquely represented by a pair of angles, (ϕ, λ) , where $0 \leq \phi \leq \pi$ is known as the *latitude* and $0 \leq \lambda \leq 2\pi$ is the *longitude*. Given two points $(\phi_1, \lambda_1), (\phi_2, \lambda_2)$ on the sphere with radius R , the geodesic distance is then derived by the spherical law of cosines:

$$\delta((\phi_1, \lambda_1), (\phi_2, \lambda_2)) = R * \arccos(\sin \phi_1 \sin \phi_2 + \cos \phi_1 \cos \phi_2 \cos(\lambda_1 - \lambda_2)) \quad (2)$$

where $\delta(X_i, X_j)$ denotes the geodesic distance between points X_i and X_j , assuming X is an $n \times 2$ matrix whose rows correspond to spherical coordinates.

It is known that the surface of a sphere cannot be perfectly preserved in any 2-dimensional Euclidean drawing, due to its curvature. One can preserve various combinations of angles, areas, geodesics, or distances but not all of these simultaneously. The orthographic projection, or the “view from space” projects the sphere onto a tangent plane with point of perspective from outside the sphere. While half of the sphere is obscured and shapes and area are distorted near the boundary, geodesics through the origin are preserved and it gives the impression of a 3-dimensional globe. The stereographic projection is similar but

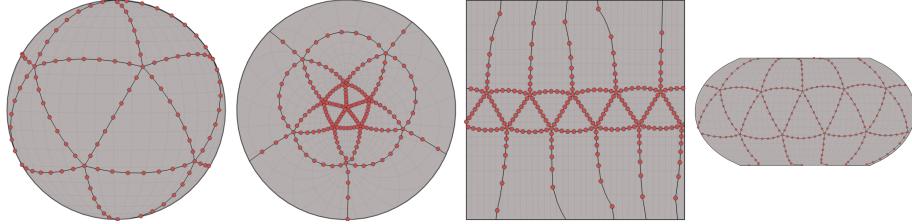


Fig. 3: A subdivided isocahedron graph embedded on the sphere, displayed with the orthographic, stereographic, Mercator, and equal Earth projections.

instead with a point of projection looking through the sphere, and preserves angles. The Mercator projection is a common cylindrical map projection with heavy area distortion near the poles. The equal Earth projection preserves area and gives the impression of a spherical shape. Examples are shown in Fig. 3. We primarily use the orthographic and equal Earth projections in this paper.

2.3 Graph Layout in non-Euclidean Geometry

Non-Euclidean graph visualization has been studied by Munzner, with an emphasis on trees and hierarchies [29,30,31,32], and the following link treevis.net, provides several examples of hyperbolic and sphere based tree visualizations [38]. Spherical layouts have been investigated in an immersive setting such as virtual reality [23,41]. Self-organizing maps have been developed for both spherical and hyperbolic geometries [33]. Several other examples of spherical graph visualization include the Map of Science [5], the “Places and Spaces” [4], and “Worldprocessor” [16] exhibitions. Some limitations of the existing algorithms for hyperbolic graph visualization are discussed in [10].

Force-directed algorithms model the nodes and edges as a physical system, and provide a layout by minimizing the total energy. These algorithms are popular in part due to their conceptual simplicity and quality layouts [18]. A general technique for generalizing force-directed algorithms to non-Euclidean spaces is described in [19]. However, it only works for small graphs as for larger ones it is too computationally expensive and is unlikely to escape local minima.

There are several different approaches to embedding a graph on the sphere. A simple idea is to generate a 2D Euclidean layout and project it onto the sphere through a linear map [8,34], however, this embedding will not make full use of spherical geometry. Another approach is to embed the graph in 3D Euclidean space and modify it to force it on the surface of a sphere [7,34], but this is quite mathematically involved and complicates the optimization. A more natural method directly computes a 2D spherical embedding (in latitude and longitude) such that the geodesic distances on the sphere and graph-theoretic distances between pairs of vertices are closely matched [9]. We focus on this approach and make it scalable by adopting stochastic gradient descent for the optimization phase and by solving the dilation/resizing problem specified below.

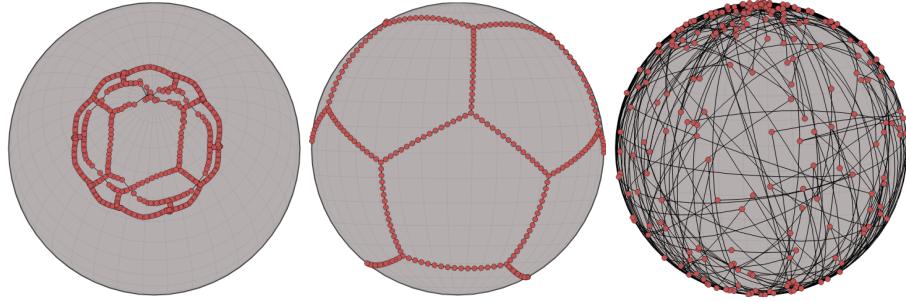


Fig. 4: A dodecahedron subdivision graph. Left: a small dilation factor forces the layout on a small patch of the sphere. Middle: a correct dilation factor using our heuristic discussed in Section 5.1, allows the graph to make use of the spherical geometry. Right: a large dilation factor makes the distances unachievable.

In the graph drawing literature, the normalized stress of a layout is a standard quality measure [12,21,43]. This is perfectly acceptable in Euclidean space where a layout is not meaningfully changed when the layout is resized. For non-Euclidean graph layouts there is a possible issue of *dilation or resizing*. Formally, a dilation is a function on a metric space M , $f : M \rightarrow M$ that satisfies $d(f(x), f(y)) = rd(x, y)$ for $x, y \in M$, $r > 0 \in \mathbb{R}$ and $d(x, y)$ being the distance between x and y . In non-Euclidean spaces, such as the sphere, the size of a layout can have drastic effects; see Fig. 4. At small dilation, a graph embedded on the sphere takes only a small patch and the sphere patch behaves like a piece of the Euclidean plane. At large dilation, a graph embedded on the sphere wraps over itself. At some optimal dilation the embedded graph fits on the sphere with low distortion. Choosing the size of the sphere is important to accurately represent the data. We are unaware of any work regarding this problem in spherical embedding, and propose a heuristic and optimization scheme to solve it in Section 5.1.

As stress is difficult to interpret between geometries, we use a more fair comparison metric called *distortion* [28,37] defined later in Section 5.

3 Algorithm

Our spherical multi-dimensional scaling algorithm (SMDS) resembles that of other stress based graph layout algorithms. That is, we first compute a graph-theoretic distance matrix via an all-pairs-shortest-paths algorithm and then use this distance matrix as an input to minimize the generalized stress function and compute vertex coordinates on the sphere. This differs from standard Euclidean MDS in that Euclidean distances between points are replaced by geodesic distances between the points on sphere. The corresponding stress function defined on the sphere is

$$\sigma_S(X) = \sum_{i < j} w_{ij} (\delta(X_i, X_j) - d_{ij})^2 \quad (3)$$

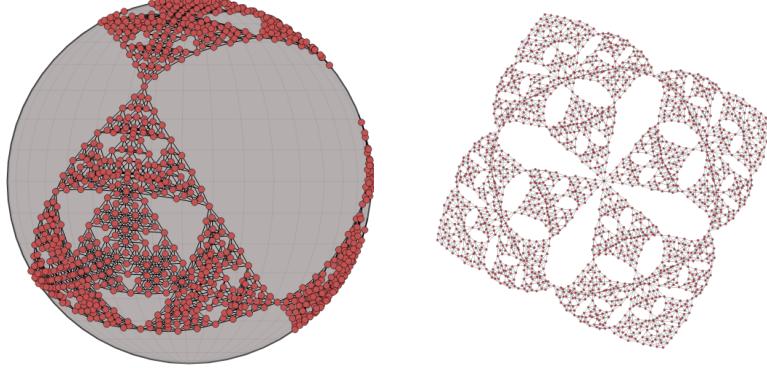


Fig. 5: The Sierpinski3d [21] graph on the sphere (left) and in the plane (right). While the Euclidean drawing on the right is aesthetically pleasing, it looks deceptively like a 2D structure and implies a center. The sphere more accurately captures the structure.

Where $\delta(X_i, X_j)$ denotes the geodesic distance between vertices i and j , d_{ij} is the graph-theoretic distance between vertex i and j , and w_{ij} is a weight, typically set to d_{ij}^{-2} . However, one can also give preferred weights based on the importance of the points and based on the application. Another typical choice is binary weights, where w_{ij} is either 0 or 1. Unless otherwise specified, δ corresponds to the geodesic distances on the unit sphere and δ_R the geodesic distances on a sphere with radius R .

We minimize equation (3) by stochastic gradient descent (SGD), which we experimentally show converges in fewer iterations while achieving lower distortion than exact gradient descent for sufficiently large graphs. SGD is a minimization approach in the gradient descent family of algorithms. Fully computing the exact gradient can be too expensive and SGD instead repeatedly performs a constant time approximation of the gradient, by considering only a single term of the sum (or subset of terms for mini-batch stochastic), which allows it to make more updates. As a result, SGD tends to converge in fewer iterations while more consistently finding the global minimum [36].

To perform SGD on the stress function, we approximate the gradient by looking at only a single pair of vertices. Note that this corresponds to one summand of the full stress function. If we rewrite equation (3) as $\sigma_S(X) = \sum_{i < j} f_{i,j}(X)$ then we can more simply write the full gradient in terms of f . Apply the chain rule to see we will need to derive the partial gradient of the geodesic distance function:

$$\begin{aligned}\frac{\partial \delta(X_i, X_j)}{\partial \phi_i} &= \frac{-\cos \phi_i \sin \phi_j - \sin \phi_i \cos \phi_j \cos(\lambda_i - \lambda_j)}{\sqrt{1 - \cos^2(\delta(X_i, X_j))}} \\ \frac{\partial \delta(X_i, X_j)}{\partial \lambda_i} &= \frac{\cos \phi_i \cos \phi_j \sin(\lambda_i - \lambda_j)}{\sqrt{1 - \cos^2(\delta(X_i, X_j))}}\end{aligned}$$

Unlike in Euclidean space, the gradient of the spherical distance function is not symmetric, i.e., $\frac{d\delta(X_i, X_j)}{d(\phi_i, \lambda_i)} \neq -\frac{d\delta(X_i, X_j)}{d(\phi_j, \lambda_j)}$. Writing out the full gradient:

$$\frac{\partial f_{i,j}}{\partial X_k} = \begin{cases} 2w_{i,j}(\delta(X_i, X_j) - d_{ij}) \left(\frac{\partial \delta(X_i, X_j)}{\partial \phi_i}, \frac{\partial \delta(X_i, X_j)}{\partial \lambda_i} \right) & k = i \\ 2w_{i,j}(\delta(X_i, X_j) - d_{ij}) \left(\frac{\partial \delta(X_i, X_j)}{\partial \phi_j}, \frac{\partial \delta(X_i, X_j)}{\partial \lambda_j} \right) & k = j \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

We can apply SGD to equation (3) by selecting pairs i, j in random order, and updating X by $X - \eta \frac{\partial f_{i,j}}{\partial X_k}$ where η is the learning rate; see Algorithm 1.

We randomly initialize the placement of vertices uniformly on the sphere, as other work has shown that SGD is consistent across initialization strategies [1,28,42].

Algorithm 1 Stochastic gradient descent algorithm for spherical MDS

```

procedure STOCHASTIC GRADIENT DESCENT(d)
     $X \leftarrow$  random initialization
    while  $\Delta(\text{stress}(X)) > \epsilon$  or max iterations is reached do
        for (i,j) in random order do
             $X_i \leftarrow X_i - \eta \frac{df_{i,j}}{dX_i}$ , according to (4)
             $X_j \leftarrow X_j - \eta \frac{df_{j,i}}{dX_j}$ , according to (4)
        end for
    end while
    return  $X$ 
end procedure

```

4 Evaluation

We first investigate the various parameters that effect SGD's optimization, then compare our results to exact gradient descent.

4.1 Hyper-parameters

There are several hyper-parameter choices to be made when using SGD. The learning rate η (also known as step size, annealing rate) has a large effect on the resulting embedding. If the learning rate is too large, the optimization will “overstep” and either fluctuate around a minimum, or diverge. If the learning rate is too small, the optimization may require many iterations to converge and is more likely to converge to a local minimum. A better strategy is to employ a learning rate schedule, where at early iterations the learning rate is large but decreases over time to allow for convergence. This is known to converge to a

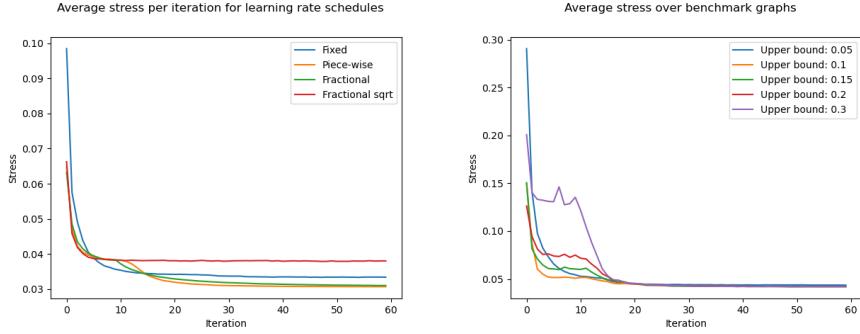


Fig. 6: (Left) Effect of the learning rate schedule on the optimization. The piece-wise schedule adapted from [42] arrives at a minimum faster on average. (Right) Effect of the upper bound on the learning rate on the optimization. An upper bound of 0.1 behaves predictably. Values for both are averaged over all graphs in our benchmark.

stationary point (could be a saddle) under certain conditions: $\sum g(t) = \infty$ and $\sum g(t)^2 < \infty$ [3].

We investigate a limited subset of possible learning rate schedules, a fixed learning rate at 0.05, a piece-wise schedule similar to that of Zheng et al. [42], a fractional decay of $\Theta(t^{-1})$, and a slower fractional decay of $\Theta(t^{-0.5})$. The piece-wise schedule begins with an exponential decay function, with large initial values and switches to $\Theta(t^{-1})$ once below a threshold. There are a few changes we needed to make to the piece-wise schedule. Firstly, while Zheng et al. [42] upper bound their learning rate by 1, this upper bound is too large for SMDS. The upper bound for the Euclidean algorithm was derived from the geometric structure, and a value of 1 reduces the stress between a single pair of vertices to 0. The latitude and longitude of the sphere are angles and so do not have this property. We instead need a relatively small upper bound, noting that large movements of a pair vertices on the sphere that need to be moved apart can actually bring them closer together (by wrapping around the sphere). We investigated values in the range 0 to 1, and settled on 0.1 as it achieves low stress quickly while not being so small as to fall into local minima; see Fig. 6.

Randomization is used to select pairs i, j in the stochastic optimization function. While SGD was originally done using sampling with replacement, random reshuffling has been shown to converge in fewer total updates [15]. To use random reshuffling in stress minimization, we enumerate all pairs $i < j$ of vertices in an ordered list and shuffle this list after every iteration. This ensures that the order in which we visit pairs is random, but that each pair is visited before we sample the same pair again.

A stopping condition is how the algorithm determines to terminate, either by converging or by reaching a maximum number of iterations. We measure convergence by tracking the change in the value of the optimization function between

iterations. In the figures and evaluation results below, we set the convergence threshold to $1e^{-7}$ or a balance between speed and quality.

4.2 Evaluation

Our code is open source and written in Python. Experiments are performed on an Intel® Core™ i7-3770 CPU @ 3.40GHz × 8 with 32 GB of RAM with a 64 bit installation of Ubuntu 20.04.3 LTS.

We use a set of 40 graphs to evaluate our SMDS algorithm: 34 from the *SuiteSparse Matrix Collection* [6] and the remaining 6 from skeletons of 3-dimensional polytopes. We use the cube, dodecahedron, and isocahedron, and subdivide them 4 times each to obtain cube_4, dodecahedron_4 and isocahedron_4. We present spherical layouts of a subset of our benchmark graphs; see Table 1. The remaining layouts can be found in the Appendix. We see that there are several graphs particularly well suited to spherical layout: the 3D polytopes and their subdivisions have much lower distortion on the sphere than in the plane. 3-dimensional meshes and triangulations of surfaces such as dwt_307 and delaunay_n10 also have lower error on the sphere.

The SGD optimization scheme performs better than exact GD on both time to convergence and stress as the size of the data becomes large as we expect; see Fig. 7.

5 Geometry Comparison

Here we discuss some possible drawbacks of graph embedding in spherical space and compare graph embeddings between Euclidean, spherical and hyperbolic spaces. Stress works well for producing layouts, but directly comparing stress scores between geometries are difficult to interpret. Layouts are often uniformly scaled so that stress is minimum before reporting (see [12,21]) which works fine in Euclidean space, but becomes a problem in spherical and hyperbolic spaces. In order to more fairly compare embedding error across geometries, we use the *distortion* [28,37] metric, defined as

$$\text{distortion}(X) = \frac{1}{\binom{|V|}{2}} \sum_{i < j} \frac{|\delta(X_i, X_j) - d_{i,j}|}{d_{i,j}}. \quad (5)$$

5.1 Dilation of Distances

It is known that Euclidean MDS is invariant to dilation, that is if one multiplies the given distances by a positive real number, the corresponding MDS solution is the original MDS solution multiplied by the same scalar factor (up to rotation). However, this is not true for spherical and hyperbolic spaces. Moreover, spherical space is bounded, unlike Euclidean space. For example, on the 2D unit sphere the maximum distance that can be achieved between two points is π (assuming that between any two points we always take the shortest geodesic distance). Any

Table 1: Layouts

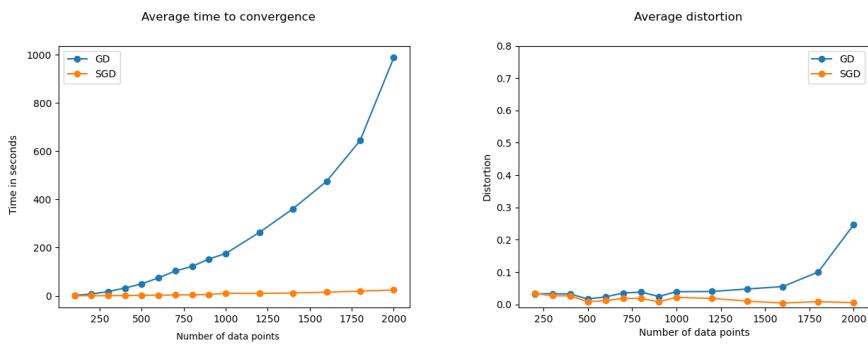
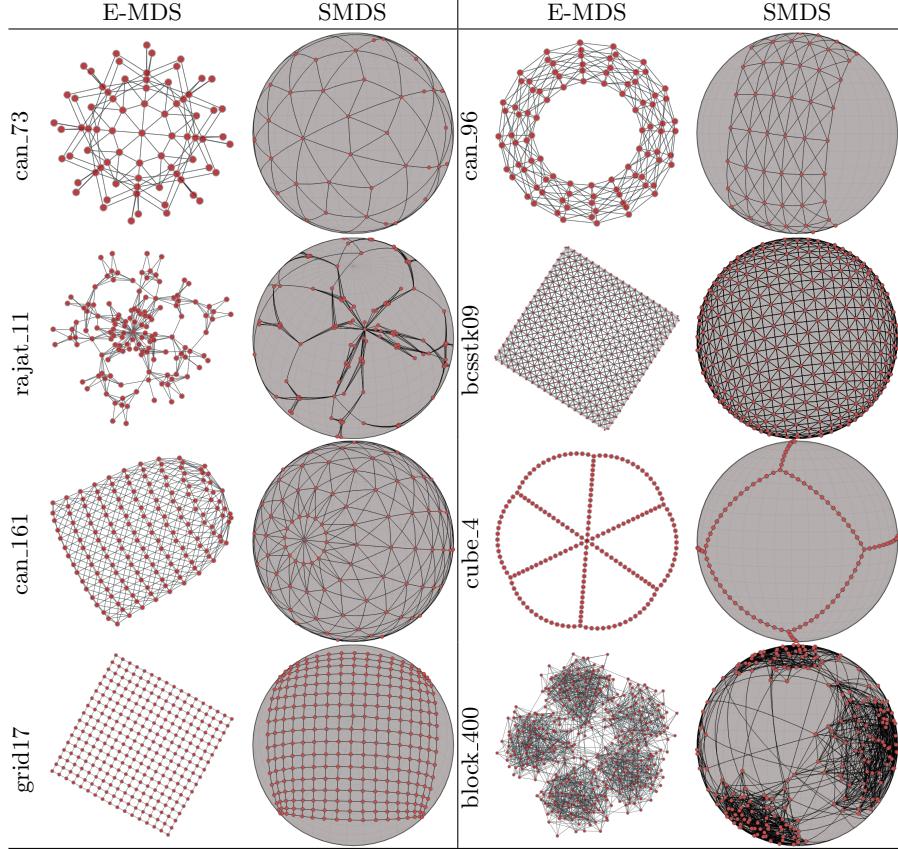


Fig. 7: How the SGD optimization scheme fairs compares to the exact GD in terms of time (left) and error (right). The larger the size of the graph, the more benefit is seen from the use of SGD.

graph with diameter (longest shortest path) longer than π cannot possibly be embedded on the unit sphere with zero error. A reasonable solution is to dilate the input distances so that all the given distances are less than or equal to π . That is, to multiply the distance matrix, d , by $\frac{\max d}{\pi}$. This heuristic appears to work well in practice; see Fig. 9. For all of our experiments and layouts, we use this heuristic. However, this has no guarantees of being optimal.

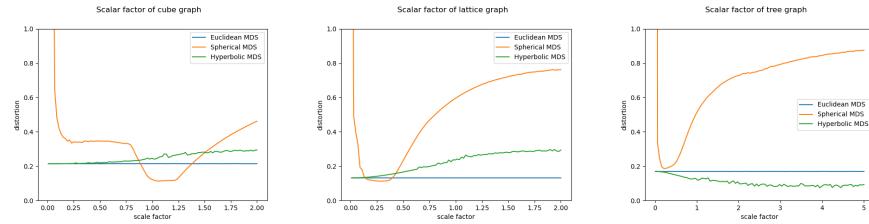


Fig. 8: Behavior of distortion on selected graphs with respect to dilation factor in each geometry.

One possible approach to the dilation problem is to make the radius of the sphere also a parameter to optimize. The problem would then become finding the best radius so that the defined stress function is as small as possible. This can be captured by reformulating Eq. (3) to also optimize the radius:

$$\arg \min_{R, X_1, \dots, X_n \in S_R^2} \sum_{i,j=1}^N (\delta_R(X_i, X_j) - d_{ij})^2. \quad (6)$$

Here $\delta_R(X_i, X_j)$ corresponds to the geodesic distance on the sphere with radius R between points X_i and X_j . We derive the gradient for R and update it along with the vertex positions at each update step.

To the best of our knowledge none of the existing algorithms for spherical embedding consider this dilation/resizing problem. However, we believe that it is a crucial parameter while embedding/drawing a graph on the sphere.

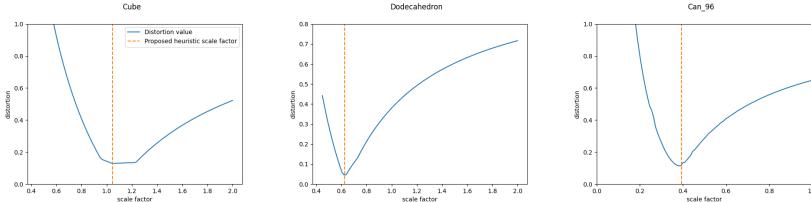


Fig. 9: Effect of dilation on distortion. Our proposed heuristic (orange line) is often very close to the minimum (of the blue curve).

5.2 Choosing Between Geometries

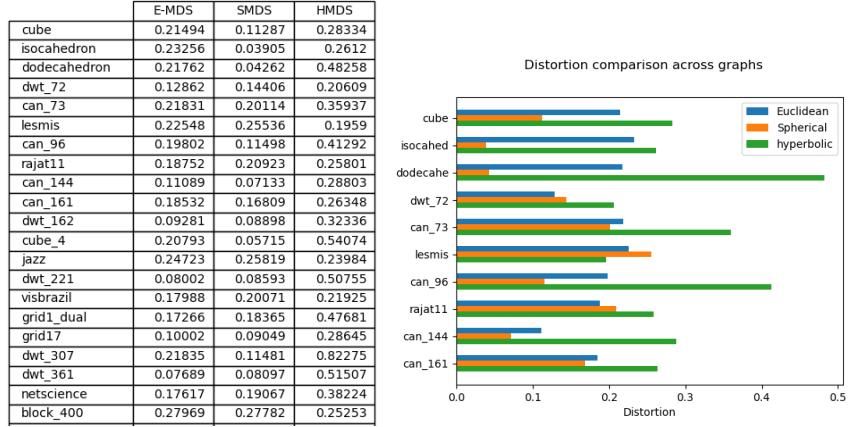


Fig. 10: The left subfigure shows a subset of results from the direct comparison for distortion in Euclidean, spherical and hyperbolic space. The right subfigure plots the first 10 rows. We note that 3D polytopes and meshes (the can graphs) are particularly well suited to the sphere, the LesMis graph is a complex network which is best embedded into hyperbolic space, and Euclidean space is better for the remaining ones.

One reason to consider embedding graphs on different manifolds (Euclidean, hyperbolic, spherical) is to be able to preserve and visualize important properties of the given graph. Some graphs achieve lower distortion on the sphere, others in hyperbolic space. In this section we investigate how spherical graph layouts differ from other consistent geometries. We choose a selection of graphs from the sparse matrix collection, and lay them each out using the Euclidean, spherical, and hyperbolic variants of MDS and measure the distortion. We repeat the layout 5 times each, and report the average distortion for each graph in each geometry. We make use of [42] for the Euclidean MDS implementation and [28] for the hyperbolic MDS (HMDS) implementation.

The hypothesis we test here is that some graphs have a dramatically lower distortion in a particular geometry. For instance, rectangular lattices can be embedded with constant error in Euclidean space [40], regular 3D polytopes can be thought of as tessellations of the sphere, and trees have been described as “discrete hyperbolic spaces” [20]. The results are summarized in Fig. 10 with additional data in Appendix B. We observe that spherical geometry is in fact able to embed polytopes and 3D meshes with lower distortion. Further, hyperbolic geometry is able to embed networks with “small-world” properties such as lesmis and block_400 with lower distortion. In graphs with 2D structure, Euclidean space is the clear winner.

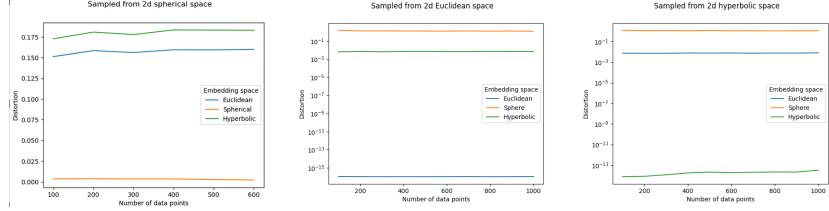


Fig. 11: Results from sampling data uniformly at random from each consistent geometry: as expected SMDS, MDS and HMDS perform dramatically better on data that comes from the geometry it embeds in.

In Fig. 11 we go beyond graphs to verify the different nature of the three geometries. We sample points randomly from each space, and use these points to define the distance matrices. We expect the corresponding geometry's MDS to embed the data with much lower distortion and this is indeed the effect we see.

6 Conclusions and Future Work

We described an efficient method for embedding graphs in spherical space. The method generalizes beyond graphs to embedding high-dimensional data. We studied (quantitatively and qualitatively) the difference between spherical embeddings of graphs and embeddings in Euclidean and hyperbolic spaces. We discussed the issue of dilation and proposed an approach that seems to work well in practice. Furthermore, we compared how structures are preserved in different geometries. The algorithm is implemented and fully functional and we provide the source code, experimental data and results, and a web based visualization tool on GitHub: <https://github.com/Mickey253/spherical-mds>.

While our proposed algorithm is much faster than exact gradient descent (5 seconds for a 1000-vertex graph), it still requires an all-pairs-shortest-paths computation as a preprocessing step, which cannot be done faster than quadratic time in the number of vertices. This is a bottleneck computation for any graph-distance based approach and coming up with a strategy (e.g., sampling a subset of distances) is a problem whose solution can impact many existing algorithms. Another direction for future work is to quickly determine the best embedding space for a given graph. That is given a graph, decide the best manifold to embed it in: Euclidean, spherical or hyperbolic. We considered stress and distortion measures here, but exploring other graph drawing aesthetics across different geometries seems to be a worthwhile direction to explore.

References

1. Börsig, K., Brandes, U., Pásztor, B.: Stochastic gradient descent works really well for stress minimization. In: Graph Drawing and Network Visualization - 28th International Symposium. vol. 12590, pp. 18–25. Springer (2020)

2. Bottou, L.: Large-scale machine learning with stochastic gradient descent. In: Proceedings of COMPSTAT 2010, pp. 177–186. Springer (2010)
3. Bottou, L.: Stochastic gradient descent tricks. In: Neural networks: Tricks of the trade, pp. 421–436. Springer (2012)
4. Börner, K.: (2012), <http://scimaps.org/home.html>
5. Börner, K., Klavans, R., Patek, M., Zoss, A.M., Biberstine, J.R., Light, R.P., Larivière, V., Boyack, K.W.: Design and update of a classification system: The ucsd map of science. *PLOS ONE* **7**(7), 1–10 (07 2012)
6. Davis, T.A., Hu, Y.: The University of Florida sparse matrix collection. *ACM Trans. Math. Softw.* **38**(1), 1:1–1:25 (2011)
7. De Leeuw, J., Mair, P.: Multidimensional scaling using majorization: SMACOF in R. *Journal of statistical software* **31**, 1–30 (2009)
8. Du, F., Cao, N., Lin, Y., Xu, P., Tong, H.: iSphere: Focus+context sphere visualization for interactive large graph exploration. In: Proceedings of the 2017 CHI Conference on Human Factors in Computing. pp. 2916–2927. ACM (2017)
9. Elbaz, A.E., Keller, Y., Kimmel, R.: Texture mapping via spherical multidimensional scaling. In: Scale Space and PDE Methods in Computer Vision. Lecture Notes in Computer Science, vol. 3459, pp. 443–455. Springer (2005)
10. Eppstein, D.: Limitations on realistic hyperbolic graph drawing. In: Purchase, H.C., Rutter, I. (eds.) *Graph Drawing and Network Visualization - 29th International Symposium, GD 2021, Tübingen, Germany, September 14-17, 2021, Revised Selected Papers*. Lecture Notes in Computer Science, vol. 12868, pp. 343–357. Springer (2021)
11. Frey, D., Pimentel, R.: Principal component analysis and factor analysis (1978)
12. Gansner, E.R., Hu, Y., North, S.C.: A maxent-stress model for graph layout. *IEEE Trans. Vis. Comput. Graph.* **19**(6), 927–940 (2013)
13. Gansner, E.R., Koren, Y., North, S.: Graph drawing by stress majorization. In: *International Symposium on Graph Drawing*. pp. 239–250. Springer (2004)
14. Gansner, E.R., Koren, Y., North, S.C.: Graph drawing by stress majorization. In: *Graph Drawing, 12th International Symposium*. Lecture Notes in Computer Science, vol. 3383, pp. 239–250. Springer (2004)
15. Gürbüzbalaban, M., Ozdaglar, A., Parrilo, P.A.: Why random reshuffling beats stochastic gradient descent. *Mathematical Programming* **186**(1), 49–84 (2021)
16. Günther, I.: (2007), <https://world-processor.com>
17. Kamada, T., Kawai, S.: An algorithm for drawing general undirected graphs. *Inf. Process. Lett.* **31**(1), 7–15 (1989)
18. Kobourov, S.: Force-directed drawing algorithms. In: *Handbook on Graph Drawing and Visualization*, pp. 383–408. Chapman and Hall/CRC (2013)
19. Kobourov, S., Wampler, K.: Non-Euclidean spring embedders. *IEEE Trans. Vis. Comput. Graph.* **11**(6), 757–767 (2005)
20. Krioukov, D., Papadopoulos, F., Kitsak, M., Vahdat, A., Boguná, M.: Hyperbolic geometry of complex networks. *Physical Review E* **82**(3), 036106 (2010)
21. Kruijger, J.F., Rauber, P.E., Martins, R.M., Kerren, A., Kobourov, S., Telea, A.C.: Graph layouts by t-sne. *Comput. Graph. Forum* **36**(3), 283–294 (2017)
22. Kruskal, J.B.: Multidimensional scaling by optimizing goodness of fit to a non-metric hypothesis. *Psychometrika* **29**(1), 1–27 (1964)
23. 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)
24. Lampert, J., Rao, R.: The hyperbolic browser: A focus + context technique for visualizing large hierarchies. *J. Vis. Lang. Comput.* **7**(1), 33–55 (1996)

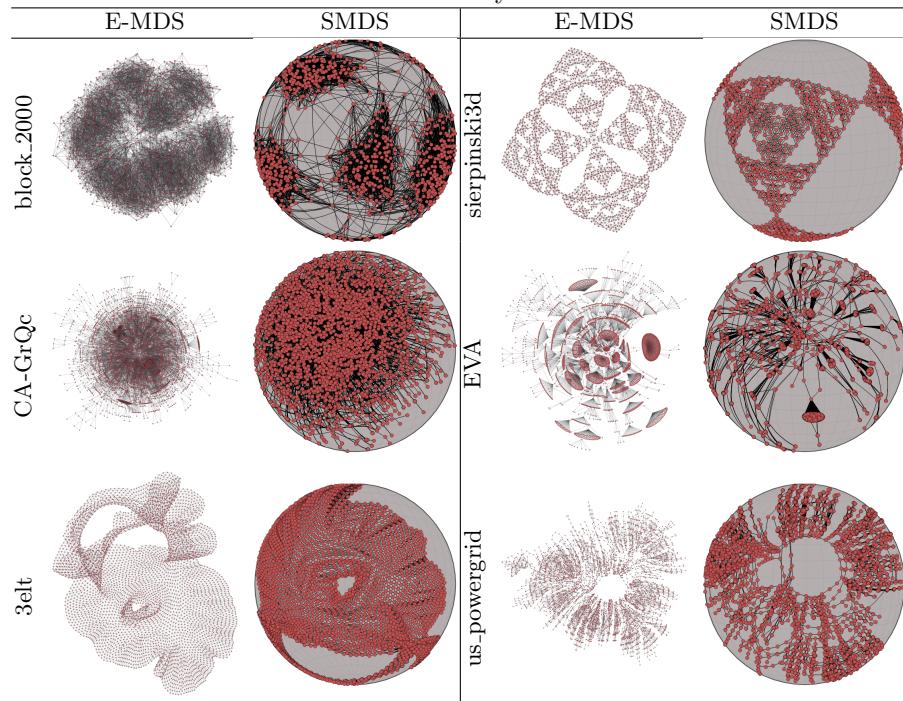
25. Lamping, J., Rao, R., Pirolli, P.: A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In: Human Factors in Computing Systems, CHI '95 Conference Proceedings. pp. 401–408. ACM/Addison-Wesley (1995)
26. Van der Maaten, L., Hinton, G.: Visualizing data using t-sne. Journal of machine learning research **9**(11) (2008)
27. McInnes, L., Healy, J.: UMAP: uniform manifold approximation and projection for dimension reduction. CoRR **abs/1802.03426** (2018)
28. Miller, J., Kobourov, S., Huroyan, V.: Browser based hyperbolic visualization of graphs. In: IEEE Pacific Visualization Symposium PacificVis. IEEE Computer Society (2022)
29. Munzner, T.: H3: laying out large directed graphs in 3d hyperbolic space. In: IEEE Symposium on Information Visualization. pp. 2–10. IEEE Computer Society (1997)
30. Munzner, T.: Exploring large graphs in 3d hyperbolic space. IEEE Computer Graphics and Applications **18**(4), 18–23 (1998)
31. Munzner, T.: Interactive visualization of large graphs and networks. Ph.D. thesis, Stanford University (2000)
32. Munzner, T., Burchard, P.: Visualizing the structure of the world wide web in 3d hyperbolic space. In: Proceedings of the 1995 Symposium on Virtual Reality Modeling Language. pp. 33–38. ACM (1995)
33. Ontrup, J., Ritter, H.J.: Hyperbolic self-organizing maps for semantic navigation. In: Advances in Neural Information Processing Systems 14. pp. 1417–1424. MIT Press (2001)
34. Perry, S., Yin, M.S., Gray, K., Kobourov, S.: Drawing graphs on the sphere. In: AVI '20: International Conference on Advanced Visual Interfaces. pp. 17:1–17:9. ACM (2020)
35. Robbins, H., Monro, S.: A stochastic approximation method. The annals of mathematical statistics pp. 400–407 (1951)
36. Ruder, S.: An overview of gradient descent optimization algorithms. CoRR **abs/1609.04747** (2016)
37. Sala, F., Sa, C.D., Gu, A., Ré, C.: Representation tradeoffs for hyperbolic embeddings. In: Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 4457–4466. PMLR (2018)
38. Schulz, H.J.: Treevis.net: A tree visualization reference. IEEE Computer Graphics and Applications **31**(6), 11–15 (2011)
39. Shepard, R.N.: The analysis of proximities: multidimensional scaling with an unknown distance function. i. Psychometrika **27**(2), 125–140 (1962)
40. Verbeek, K., Suri, S.: Metric embedding, hyperbolic space, and social networks. Comput. Geom. **59**, 1–12 (2016)
41. Yang, Y., Jenny, B., Dwyer, T., Marriott, K., Chen, H., Cordeil, M.: Maps and globes in virtual reality. Comput. Graph. Forum **37**(3), 427–438 (2018)
42. Zheng, J.X., Pawar, S., Goodman, D.F.M.: Graph drawing by stochastic gradient descent. IEEE Trans. Vis. Comput. Graph. **25**(9), 2738–2748 (2019)
43. Zhu, M., Chen, W., Hu, Y., Hou, Y., Liu, L., Zhang, K.: DRGraph: An efficient graph layout algorithm for large-scale graphs by dimensionality reduction. IEEE Trans. Vis. Comput. Graph. **27**(2), 1666–1676 (2021)

Appendix

A Additional Layouts

We present additional layouts on the Euclidean plane and on the sphere.

Table 2: Layouts



B Further geometry comparison

	E-MDS	SMDS	HMDS
cube	0.21494	0.11287	0.28334
isocahedron	0.23256	0.03905	0.2612
dodecahedron	0.21762	0.04262	0.48258
dwt_72	0.12862	0.14406	0.20609
can_73	0.21831	0.20114	0.35937
lesmis	0.22548	0.25536	0.1959
can_96	0.19802	0.11498	0.41292
rajat11	0.18752	0.20923	0.25801
can_144	0.11089	0.07133	0.28803
can_161	0.18532	0.16809	0.26348
dwt_162	0.09281	0.08898	0.32336
cube_4	0.20793	0.05715	0.54074
jazz	0.24723	0.25819	0.23984
dwt_221	0.08002	0.08593	0.50755
visbrazil	0.17988	0.20071	0.21925
grid1_dual	0.17266	0.18365	0.47681
grid17	0.10002	0.09049	0.28645
dwt_307	0.21835	0.11481	0.82275
dwt_361	0.07689	0.08097	0.51507
netscience	0.17617	0.19067	0.38224
block_400	0.27969	0.27782	0.25253
dwt_419	0.08344	0.07366	0.2256
oscil_dcop_01	0.11012	0.13286	0.39312
can_445	0.20679	0.23389	0.5722
isocahedron_4	0.22784	0.1209	0.38979
dodecahedron_4	0.2085	0.05359	0.49488
494_bus	0.19466	0.21864	0.92698
dwt_918	0.10219	0.11721	0.3245
price_1000	0.23836	0.26118	0.52257
dwt_1005	0.10518	0.08336	0.79854
cage8	0.26613	0.27013	0.38178
delaunay_n10	0.20124	0.13944	0.6588
bcsstk09	0.09826	0.13765	1.48744
fpga	0.22892	0.25561	0.45647
block_2000	0.29948	0.30314	0.27941
sierpinski3d	0.19641	0.2416	0.70151
CA-GrQc	0.27457	0.28748	0.48946
EVA	0.2416	0.26602	0.49428
3elt	0.14142	0.15756	0.85245
us_powergrid	0.17774	0.19339	0.78793

Fig. 12: Average distortion values on each graph for each geometry.

Table 3: Layouts

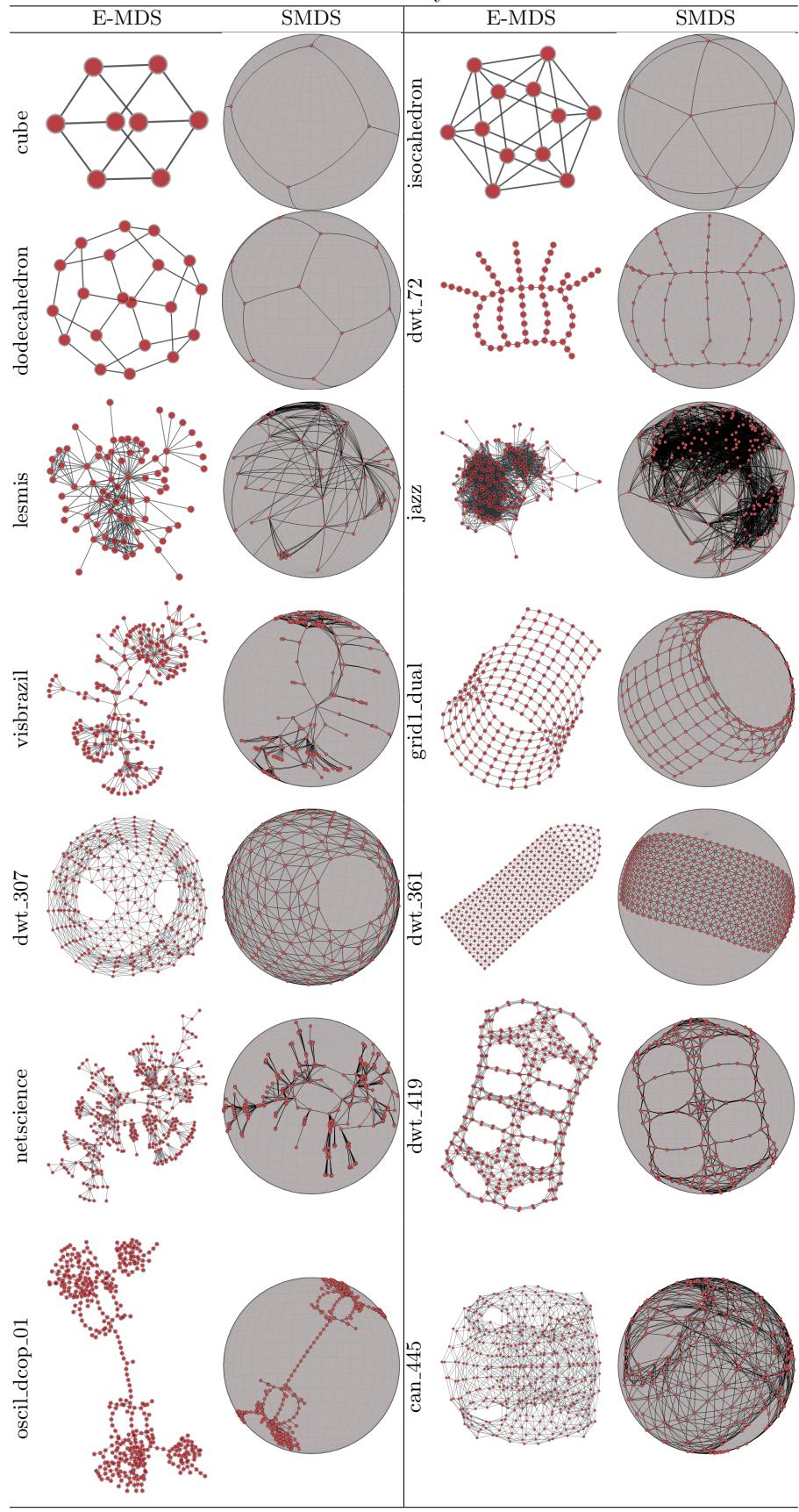


Table 4: Layouts

