Force-Directed 3D Arc Diagrams

Michael J. Bannister, Michael T. Goodrich, Peter Sampson

University of California, Irvine, USA

Abstract. We discuss a force-directed algorithm for constructing 3D arc diagrams. We introduce forces that allow curves in a 2D force directed graph to bow out and away from each other in the third dimension in order to achieve better angular resolution.

1 Introduction

In a 2D *arc diagram*[1], a graph is drawn by placing its vertices on a line and drawing its edges as circular arcs. Goodrich and Pszona [4] extend this definition to 3D arc diagrams, where vertices are placed in the xy-plane and edges are drawn as circular arcs that may bow out of that plane in the third dimension, and they show how to use graph coloring methods to determine tangent angles. Their approach works well for some types of graphs, but not all. In this poster, we present a general force-directed method for constructing a 3D arc diagram for any graph.

2 Our Algorithm

We start with a 2D force-directed layout produced with Fruchterman-Reingold [3] forces, that is, where all vertices repel each other based on electromagnetic forces and vertices connected by an edge are attracted to each other by a force that views the edge as a spring. Vertex placements are refined iteratively, where each iteration brings the 2D graph closer to a low energy state where nodes are experiencing the same forces in every direction. Our implementation starts with the standard 2D forces, without any kind of distortion, allowing vertices to move freely. Thus, in the xy-plane, all vertices exert a force inversely related to their distance, similar to an electromagnetic force, which tends to push away vertices that are not related to each other. Vertices that share an edge are pulled toward each other with a spring force, positively related to their distance, so that nodes that are related can be closer spatially. These two forces are tempered by a cooling function that gradually reduces the impact of the forces to avoid any case where nodes are pulled and forth indefinitely. This phase of the algorithm is similar to the first phase in the "dummy node" approach of Chemobelskiy *et al.* [2] for force-directed Lombardi drawings.

After a small number of iterations, our algorithm enters its second phase, where we allow intersecting edges to "pop" out of the *xy*-plane. We also allow edges that share common node to bow away from each other in the third dimension, based on their

[1] See `http://en.wikipedia.org/wiki/Arc_diagram`.

proximity and the angle between them. This is implemented by viewing edges as Bezier curves with 1 control point. The bowing force is exerted on the control points of two edges sharing a common node and is constructed by the current distance between the control points and the angle formed by the shared node and the location of each control point. Specifically, we compute a force that 2 control points will exert on each other as

$$F = c/(r^a \cdot d^b),$$

where r is the angle of the control points and their shared node and d is the distance between the two control points. In our case, we found that the values, $c = .5$ and $b = 2$, worked well, with $r = 1.5$ initially.

This force is applied in the x- and y-directions, leading the two control points away from each other and in the positive z-direction on both. It is then reduced so that the xy-force is perpendicular to the original straight line edge before any curve bowing force is applied. The control points are then pulled back to their original, unmoved location laying on the *xy*-plane by a spring force with power proportional to how far away the control point currently is from its origin. The spring force for a direction (x in this case) is given by

$$-Fx = kFx,$$

where $k = .5$.

When edges intersect, as detected by an orientation algorithm [1], we repel their control points in the z-direction only. This lifts up the higher edge and lowers the other. The magnitude of the force exerted on each control point is based on their distance in the *xy*-plane.

In the initial state, where all edges are lying on the *xy*-plane and have a z-value of 0, we exert a force to only one of the edges (at random), leaving the other edge still on the *xy*-plane.

3 Conclusion

We have observed that by adding another dimension to a graph drawing and allowing control points to enter this new dimension with forces that cause edges to bow out of the plane in a fashion dictated by intersections, angles, and proximity, we can improve the angular resolution of the graph as a whole. For almost all cases, using additional forces will provide better angular resolution than if we were in the *xy*-plane alone.

References

1. G. D. Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. 1998.
2. C. A. Duncan, D. Eppstein, M. T. Goodrich, S. G. Kobourov, and M. Nöllenburg. Lombardi drawings of graphs. *Graph Drawing*, pp. 195–207. Springer, LNCS 6502, 2011, doi:10.1007/978-3-642-18469-7_18.
3. T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience* 21(11):1129–1164, 1991, doi:10.1002/spe.4380211102.
4. M. T. Goodrich and P. Pszona. Achieving good angular resolution in 3d arc diagrams. *Graph Drawing*, pp. 161–172. Springer, LNCS 8242, 2013, doi:10.1007/978-3-319-03841-4_15.

Force-Directed 3D Arc Diagrams

Michael J. Bannister Michael T. Goodrich Peter Sampson
University of California, Irvine

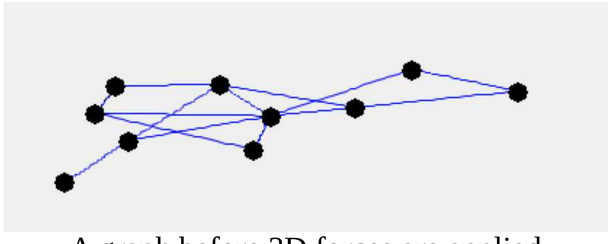
Abstract

A 3D arc diagram draws vertices in the *xy*-plane and bows edges out of the plane as circular arcs so as to provide better angular resolution. We give a force-directed method for producing a 3D arc diagram of any graph, G, so as to take edge overlaps into account, whereas previous approaches [4] ignored edge overlaps.

Method

Our approach is to have the forces on the *xy*-plane work independently from forces pushing edges out of the plane. The 2D forces are Fruchterman-Reingold forces where nodes repel each other with similar to electromagnetic particles based on Coulomb's law and nodes connected with an edge are attracted with spring forces based on Hooke's law.

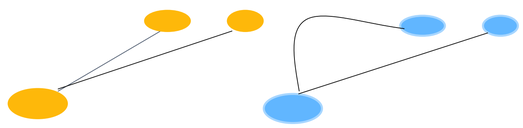
Our forces that push the edges out of the *xy*-plane work similarly by using a dummy node placed in the center of the edge and these dummy nodes are pushed up and away from each other in order to bow edges sharing a node away from each other. There is also a spring force that pulls these dummy nodes and therefore edges back to their initial state.



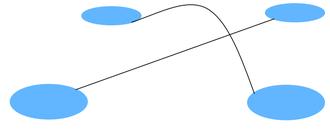
A graph before 3D forces are applied

3D Forces

Curve Bowing – lifts up an edge by its dummy node (center control point on a bezier curve) and pushes it away from the control point of a different edge on a shared node. This force is strongest on edges coming from a shared node that are too close in proximity or the angle between them is too small. This force increases the angular resolution of the graph as a whole by increasing the minimum angles of the edges of the graph.



Intersection Lifting – Using an orientation algorithm to determine if edges intersect, this will vertically push the dummy nodes away from each other, improving the visible of the graph and removing the intersection. This will result in one edge arcing over the other, rather than crossing it.



Spring Forces – Attempt to pull the dummy nodes back to their initial state and will work directly against the Curve Bowing force to find the best angular resolution.

Results

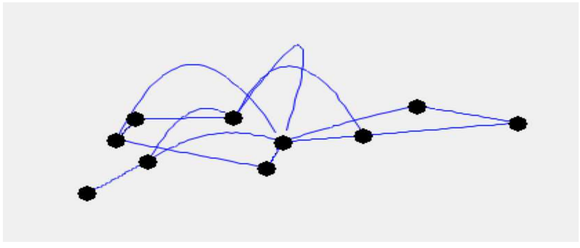
Below are the angular resolutions of several graphs (Grafo200-209 are Rome graphs), with the 3D forces enabled and with them disabled. The graphs range from 8 nodes (Cube) to 38 nodes (Grafo208). The angular resolution is measured as the smallest angle between the tangents of two edges.

Graph	With 3D forces	Only 2D forces
Petersen	80.74	49.09
Cube	32.61	25.60
Grafo200	25.79	24.04
Grafo201	42.54	38.59
Grafo202	85.97	48.61
Grafo203	92.02	71.78
Grafo204	130.98	117.12
Grafo205	53.10	56.68
Grafo206	136.93	133.45
Grafo207	70.06	47.14
Grafo208	18.09	19.34
Grafo209	146.60	128.98

In some cases, the angular resolution without the 3D forces is superior (Grafo205, Grafo208). This is due to the fact that intersection lifting can reduce the angular resolution but provides the graph more clarity as a result.

Conclusion

The data shows that for some graphs, using 3D forces can greatly improve angular resolution and the forces work such that if angular resolution cannot be improved, the forces will not act on the edges. Different graph orientations yield different results, a densely packed graph will have more room for improvement than one where nodes and edges are sparse. When they are sparse, the 2D forces will do most of the work making additional 3D forces unnecessary. In a dense graph, 3D forces can in some cases nearly double the angular resolution.



A graph after 3D forces are applied

References

1. G. D. Battista, P. Eades, R. Tamassia, and I. G. Tollis. Graph Drawing: Algorithms for the Visualization of Graphs. 1998.

2. R. Chernobelskiy, K. I. Cunningham, M. T. Goodrich, S. G. Kobourov, and L. Trott. Force-directed lombardi-style graph drawing. Graph Drawing, pp. 320–331. Springer Berlin Heidelberg, LNCS 7034, 2012, doi:10.1007/978-3-642-25878-7 31.

3. T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. Software: Practice and Experience 21(11):1129–1164, 1991, doi:10.1002/spe.4380211102.

4. M. T. Goodrich and P. Pszona. Achieving good angular resolution in 3d arc diagrams. Graph Drawing, pp. 161–172. Springer, LNCS 8242, 2013, doi:10.1007/978-3-319-03841-4 15.

Contact Information

mbannist@uci.edu
goodrich@acm.org
petersampson32@gmail.com