MTTTS17
Dimensionality Reduction and Visualization

Spring 2020
Jaakko Peltonen

Lecture 13: Visualization and navigation of graphs - Graph Layout

Slides originally by Francesco Corona and Manuel J.A. Eugster

Part 5:
Layout for general-structured graphs

How to cure the Hairball?

Possible parts to play with:

- Visual attributes of edges and vertices
- **Layout algorithm** **<---- we will concentrate on this**
- Edge bundling
- **Interaction** **<---- we will also talk about this**

Graph drawing: Optimization algorithms

Usually, a graph has infinitely many different drawings. However, the usefulness of a drawing of a graph depends on its readability.

Readability issues are expressed by means of aesthetics, which can be formulated as optimization goals for the drawing algorithms.

A fundamental and classical aesthetic is the minimization of **crossings between edges**.

(*) cf. Battista et al. (1994)

Graph drawing: Optimization algorithms

Minimization of edge crossings is a difficult goal to use, as crossings depend in complicated ways on the node placement.

Popular graph layout strategies use alternative approaches:

- force-based
- circular
- tree-based

Force-Directed layout algorithms

Force-directed methods define an objective function which maps each graph layout into a number in $\mathbb{R}^+$ representing the energy of the layout.

This function is defined in such a way that low energies correspond to layouts in which adjacent nodes are near some pre-specified distance from each other, and in which non-adjacent nodes are well-spaced. A layout for a graph is then calculated by finding a minimum of this objective function.

(*) Tamassia (2013, Chapter 12)

Kamada-Kawai Method

Tomihisa Kamada and Satoru Kawai. An Algorithm for Drawing General Undirected Graphs. Information Processing Letters 31:7-15, 1989.

Imagine that the *n* vertices in an on-screen graph are connected by **springs**. The graph becomes a dynamical system that tries to reach a minimum-energy state, where springs are, as much as possible, not stretched or compressed overmuch from their “relaxed-state length”.

$$Cost = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{2} k_{ij} \left(\| \mathbf{y}_i - \mathbf{y}_j \| - l_{ij} \right)^2$$

$$l_{ij} = L \cdot d_{ij}$$

desired on-screen length of edge (i,j)

length of shortest path (n. of steps) from i to j

desired on-screen length of an edge

$$k_{ij} = K / d_{ij}^2$$

constant

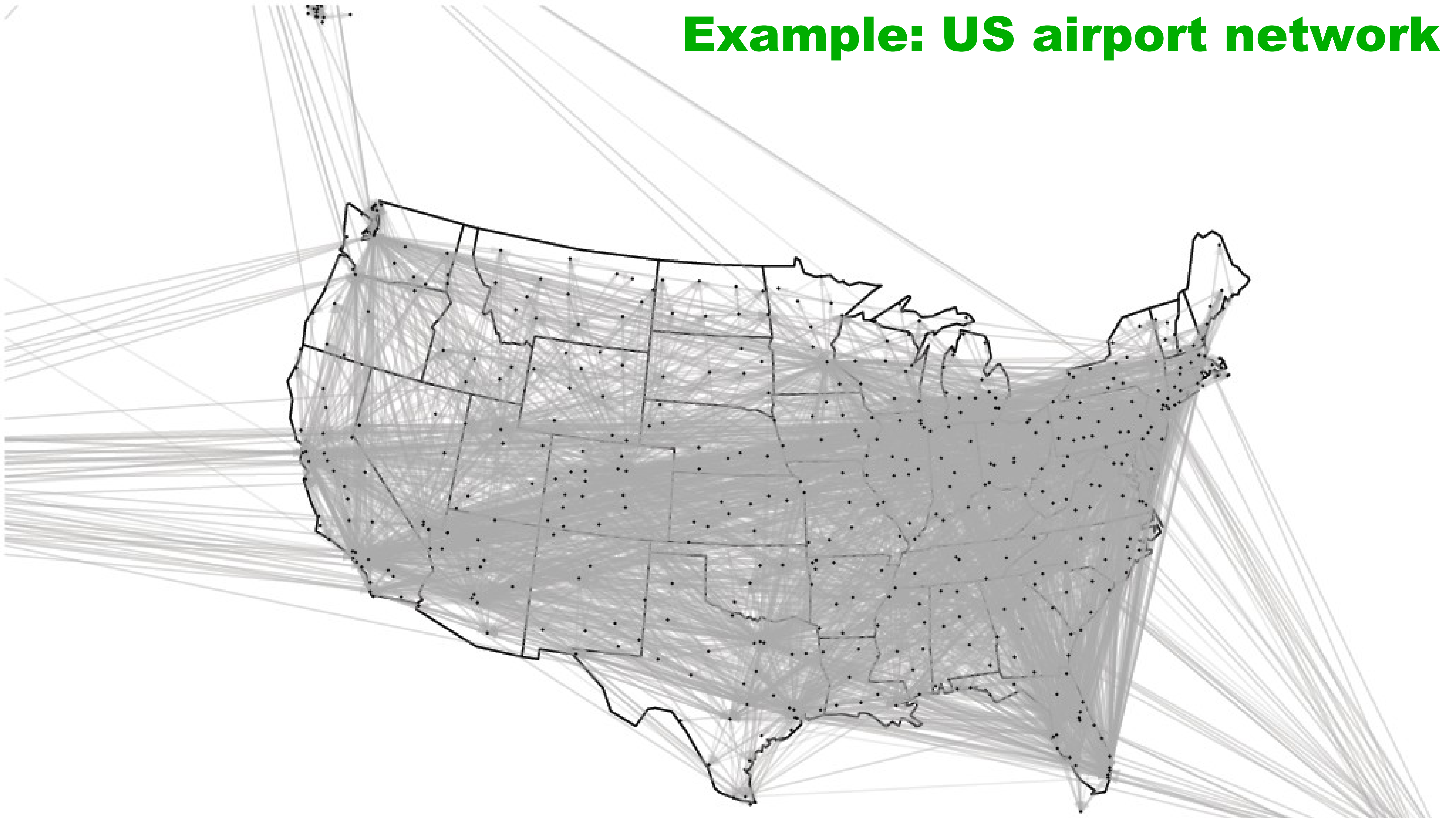
strength of spring (i,j)

Use nonlinear optimization (gradient descent or the Newton-Raphson method) to minimize the energy with respect to the $\mathbf{y}_i$

Gradients = Like the springs are pushing and pulling at the nodes

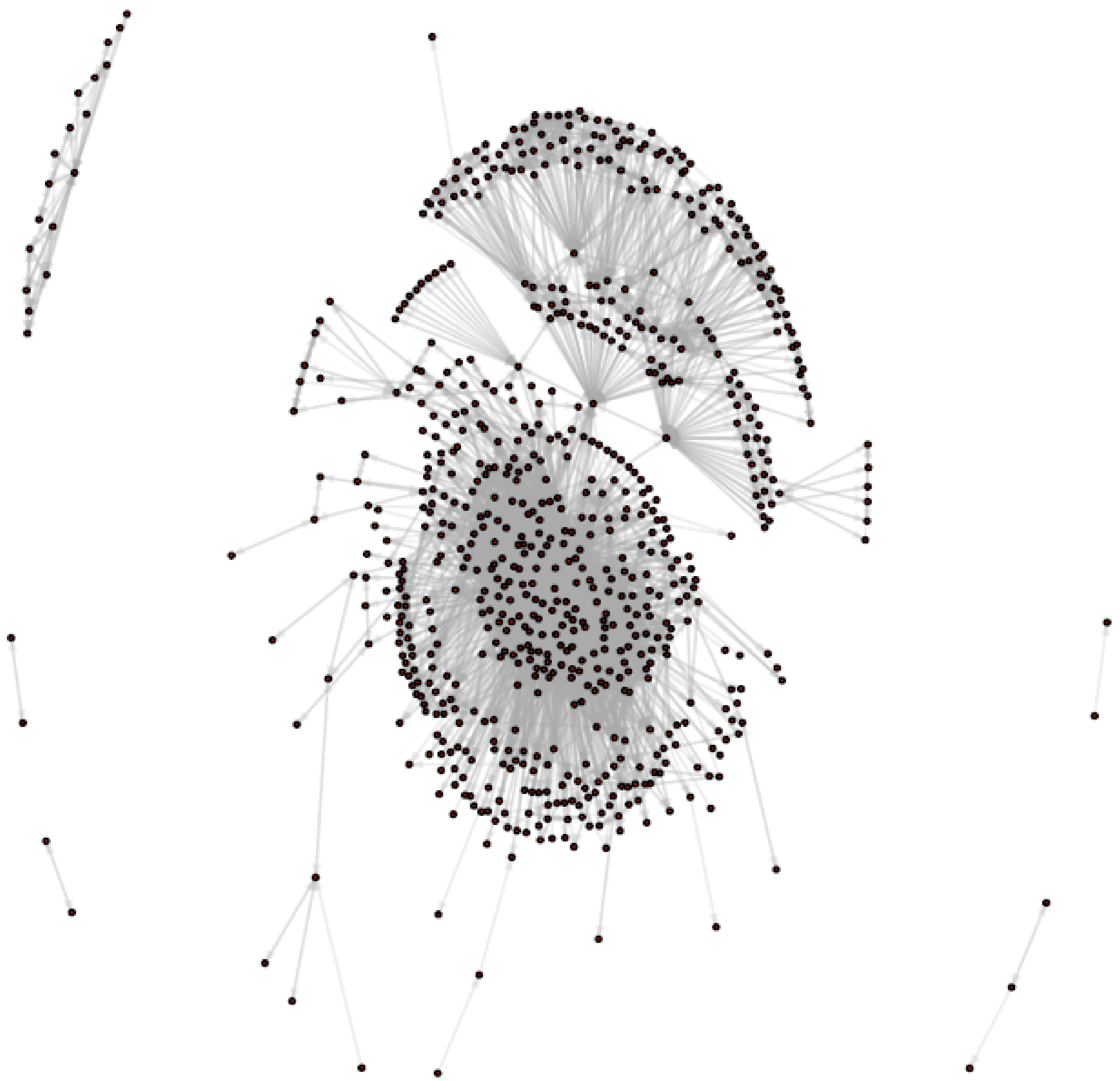
Graph drawing with fixed node locations

Example: US airport network



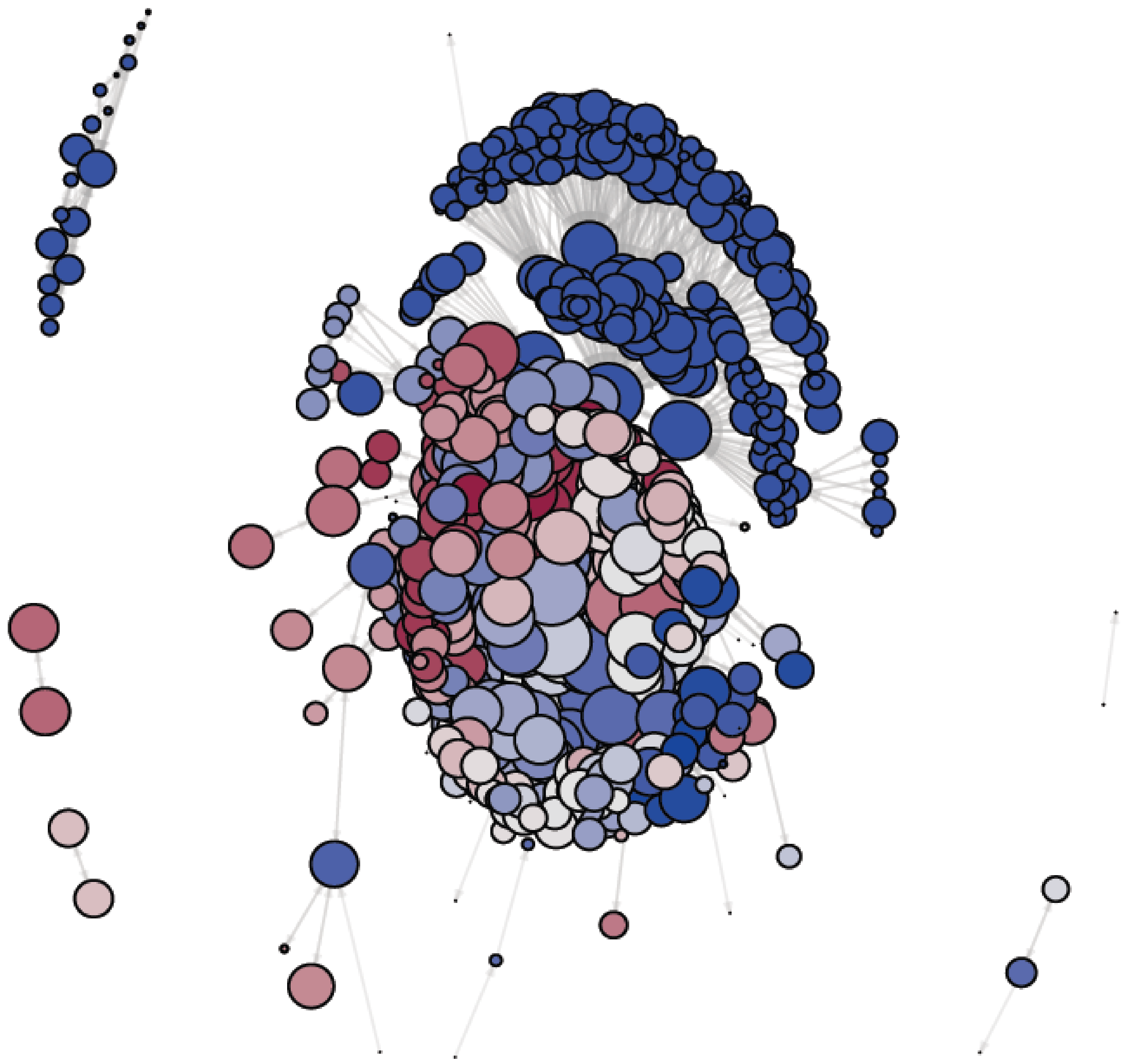
Airways.

US airport network



Kamada-Kawai layout with weighted edges based on the distance.

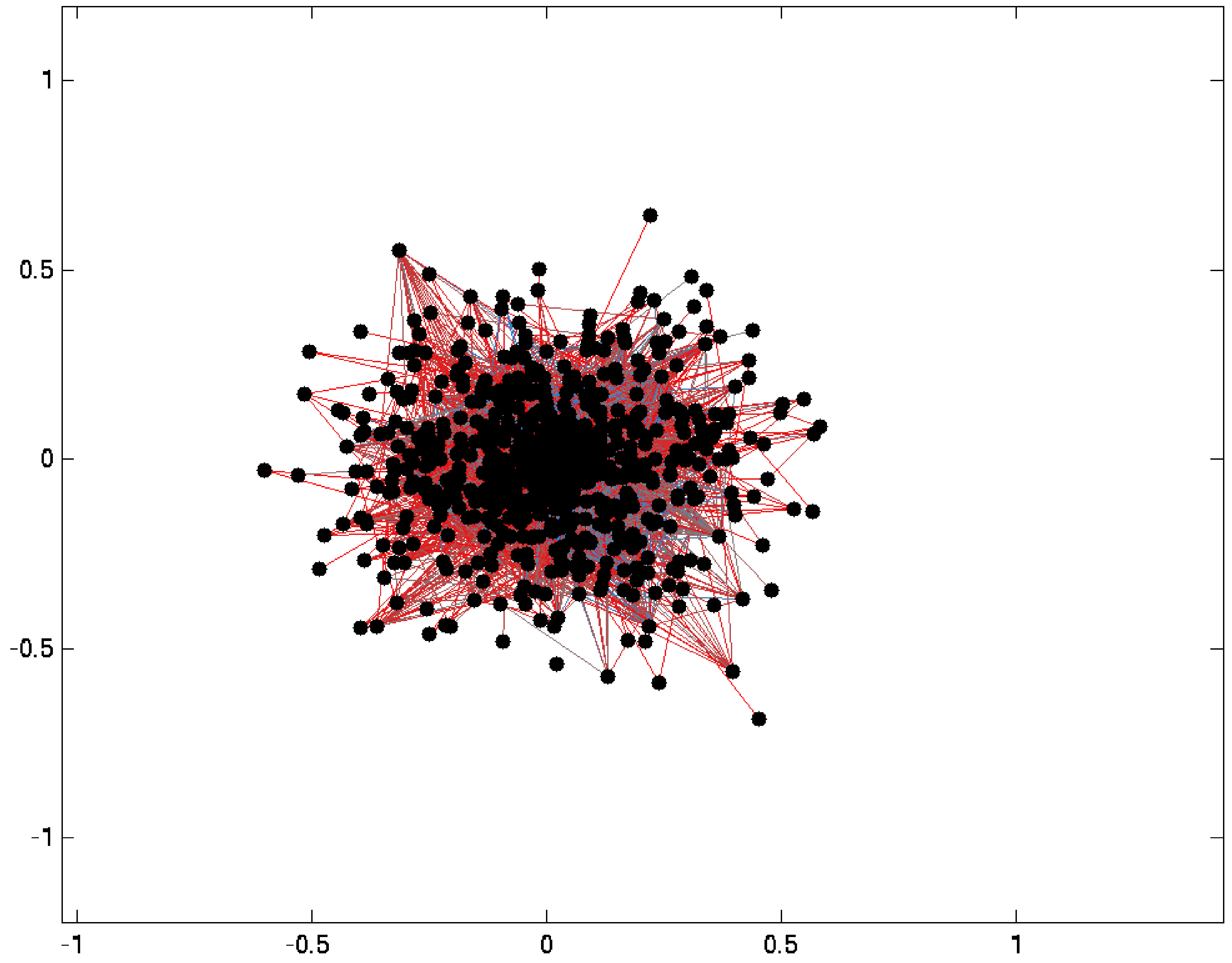
US airport network



Kamada-Kawai layout with weighted edges based on the distance; vertex sizes based on total passengers, vertex color based on state.

US airport network

In this visualization, edges are drawn red if they are too long compared to the desired length

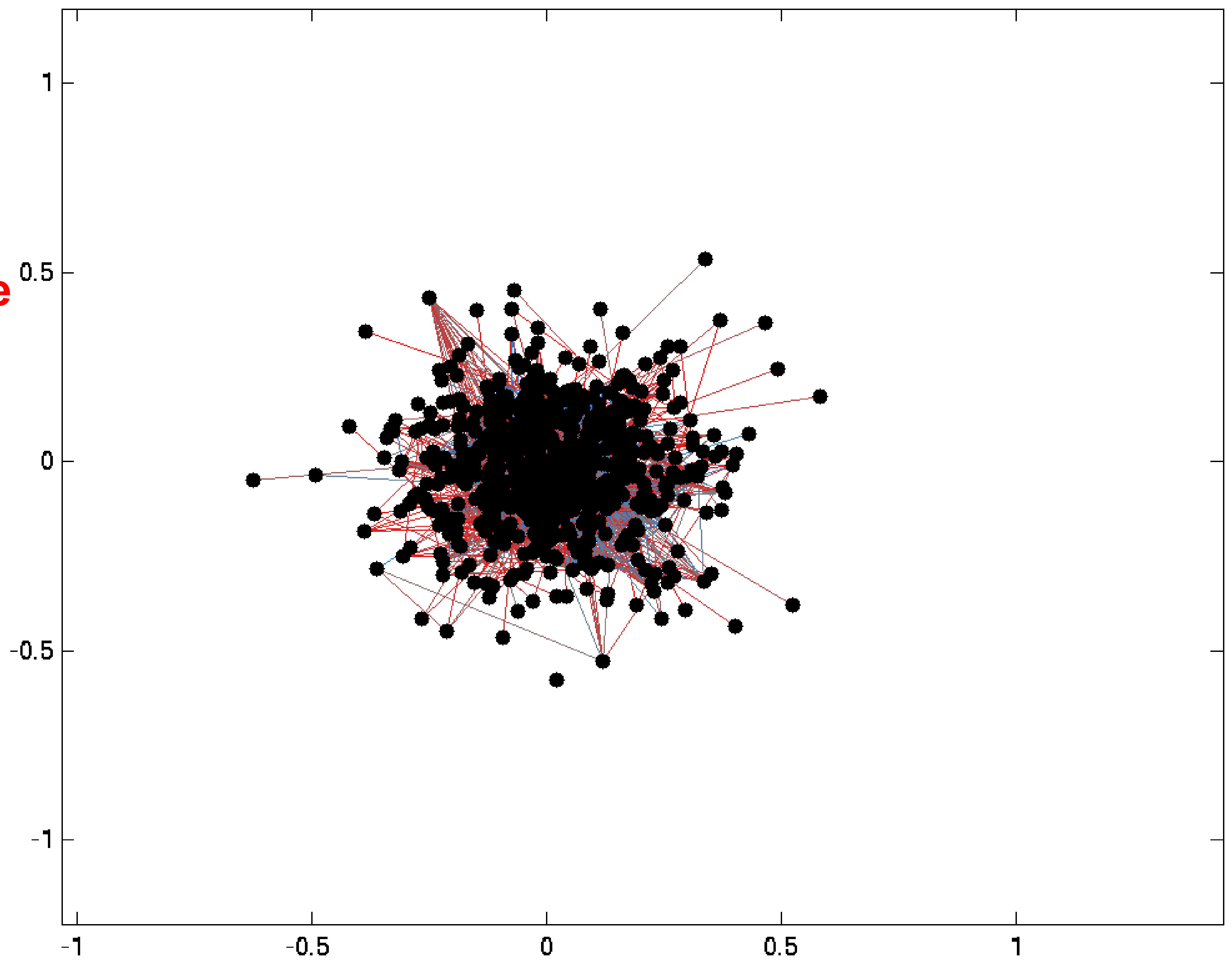


In this visualization, edges are drawn blueish if they are too short compared to the desired length

Iteratively minimizing the Kamada-Kawai cost function with weighted edges based on the distance: 1*500 iterations

US airport network

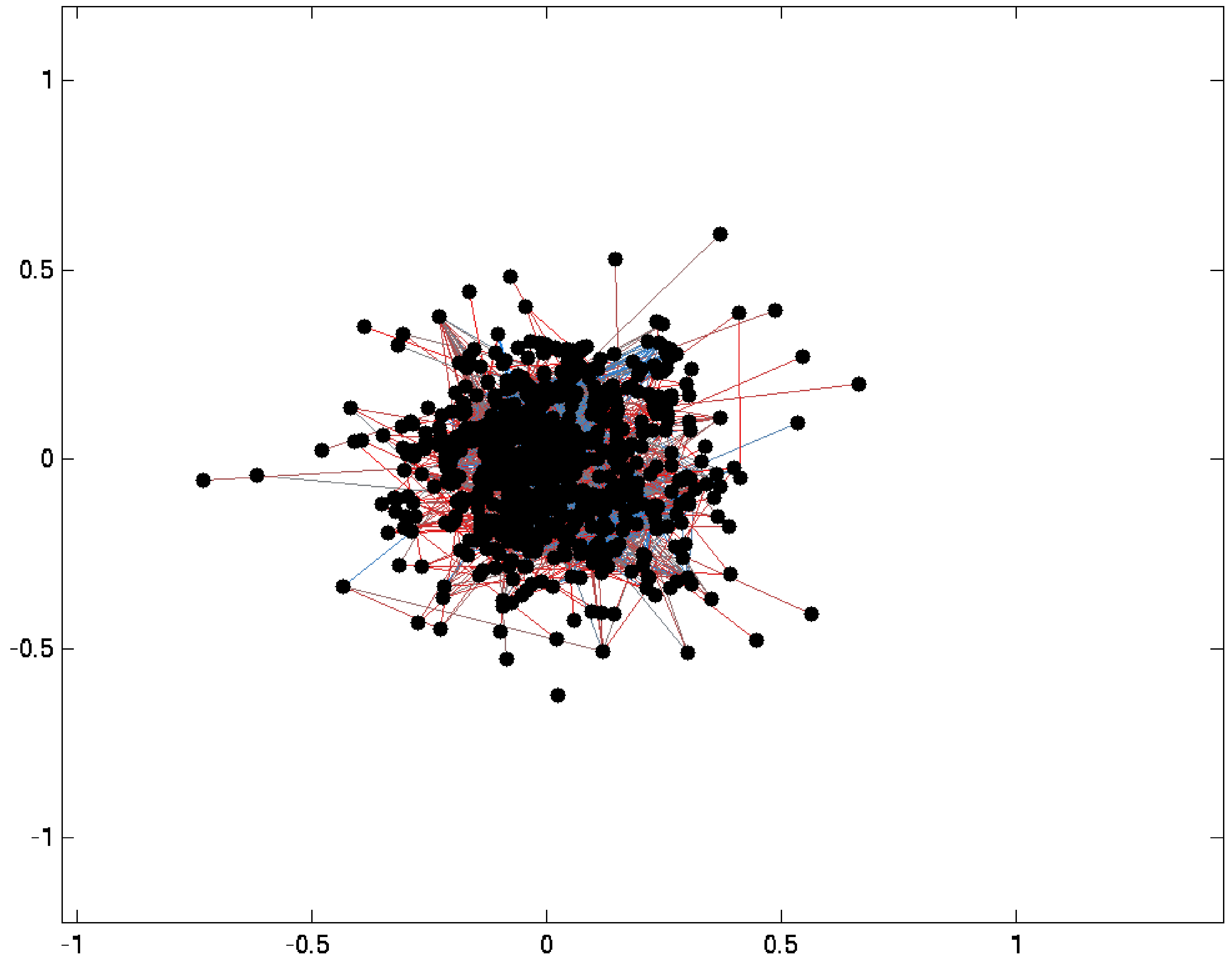
At first, the many too-long edges cause the nodes to be drawn towards the center



Iteratively minimizing the Kamada-Kawai cost function with weighted edges based on the distance: 5*500 iterations

US airport network

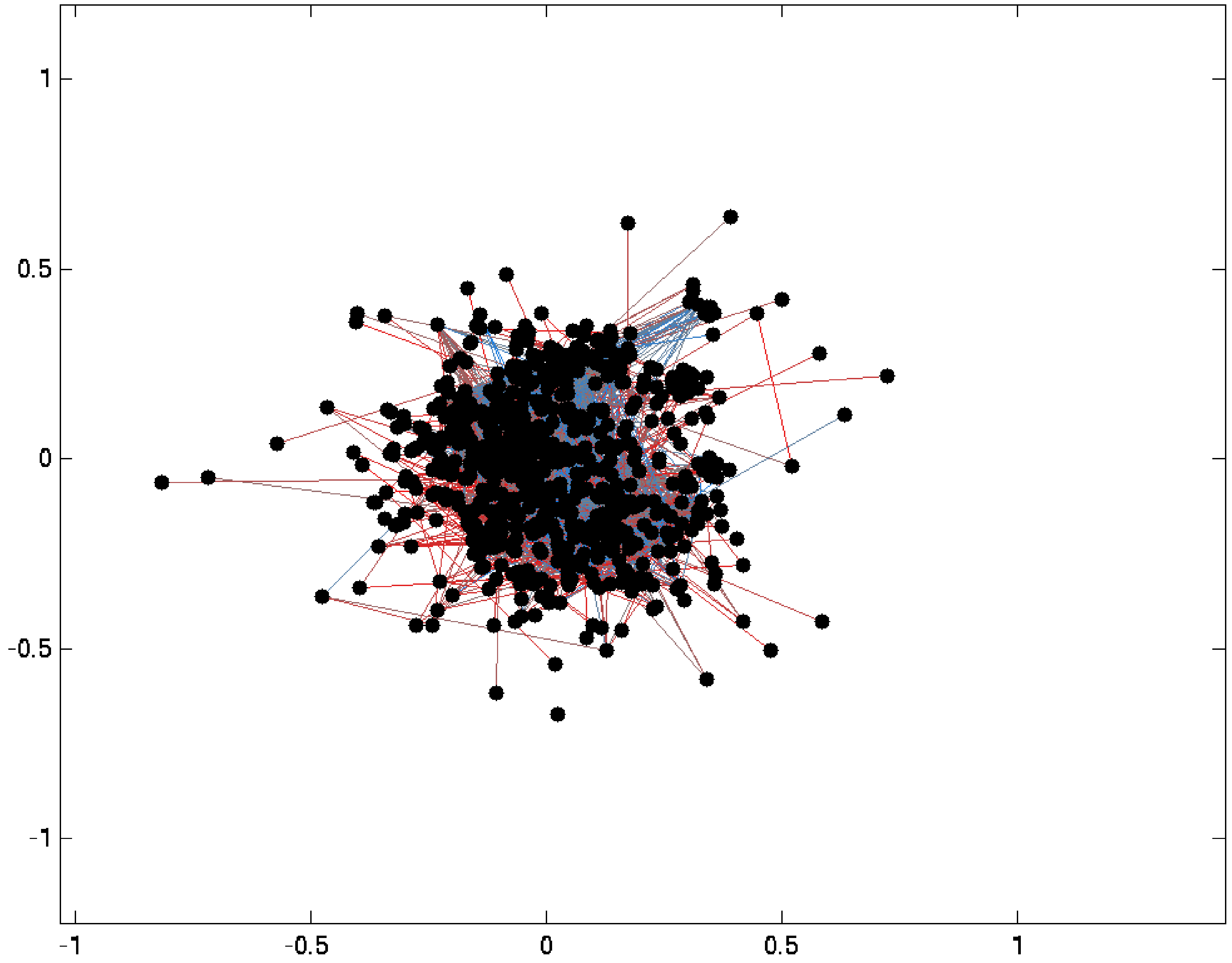
Then the too-short edges start to push away from the central area



Iteratively minimizing the Kamada-Kawai cost function with weighted edges based on the distance: 10*500 iterations

US airport network

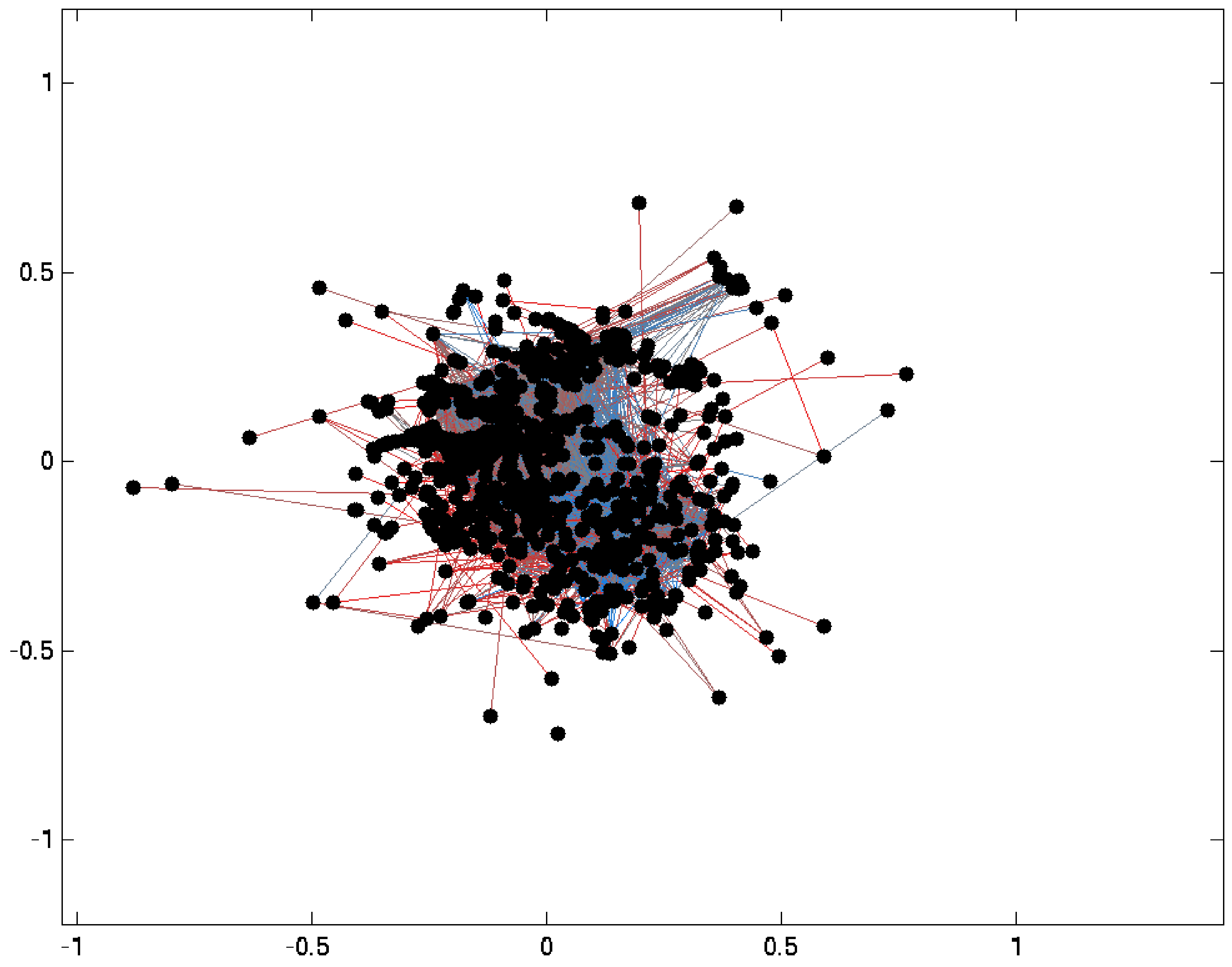
Then the too-short edges start to push away from the central area



Iteratively minimizing the Kamada-Kawai cost function with weighted edges based on the distance: 15*500 iterations

US airport network

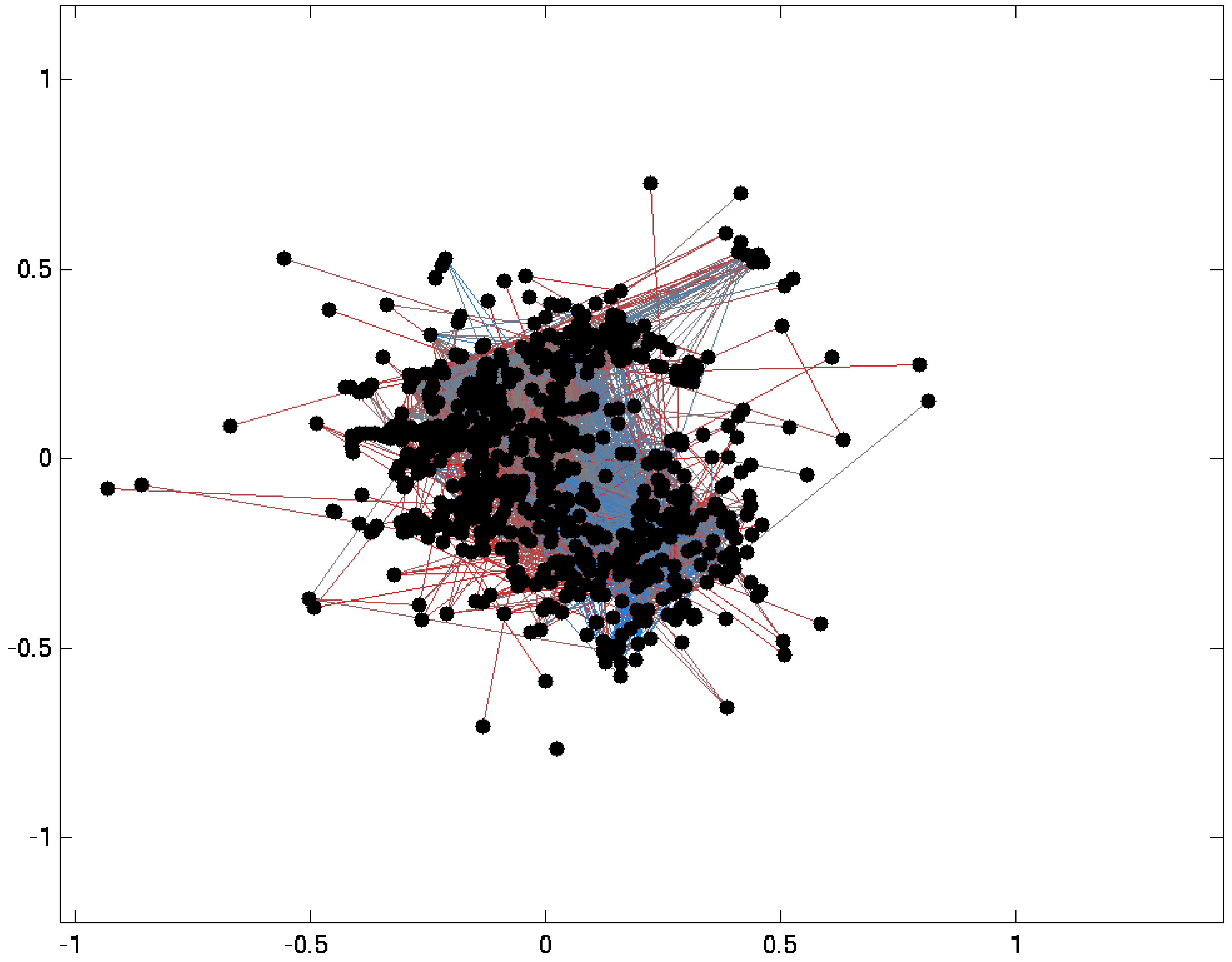
Then the too-short edges start to push away from the central area



Iteratively minimizing the Kamada-Kawai cost function with weighted edges based on the distance: 20*500 iterations

US airport network

Then the too-short edges start to push away from the central area

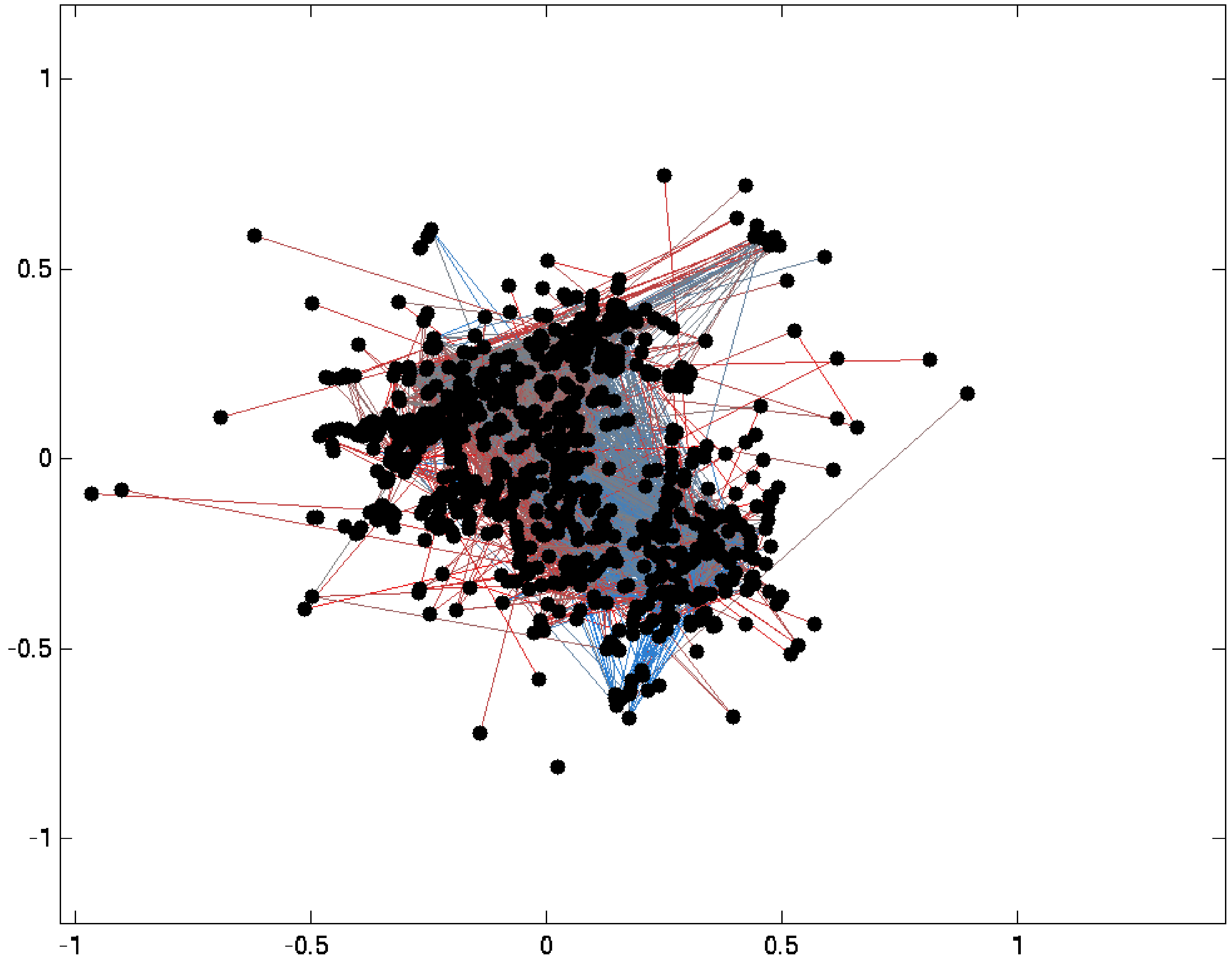


Iteratively minimizing the Kamada-Kawai cost function with weighted edges based on the distance: 25*500 iterations

US airport network

The system tries to find a balance between the “springs”

---> vertices whose edges have small desired distance tend to move in groups away from the more dissimilar vertices.

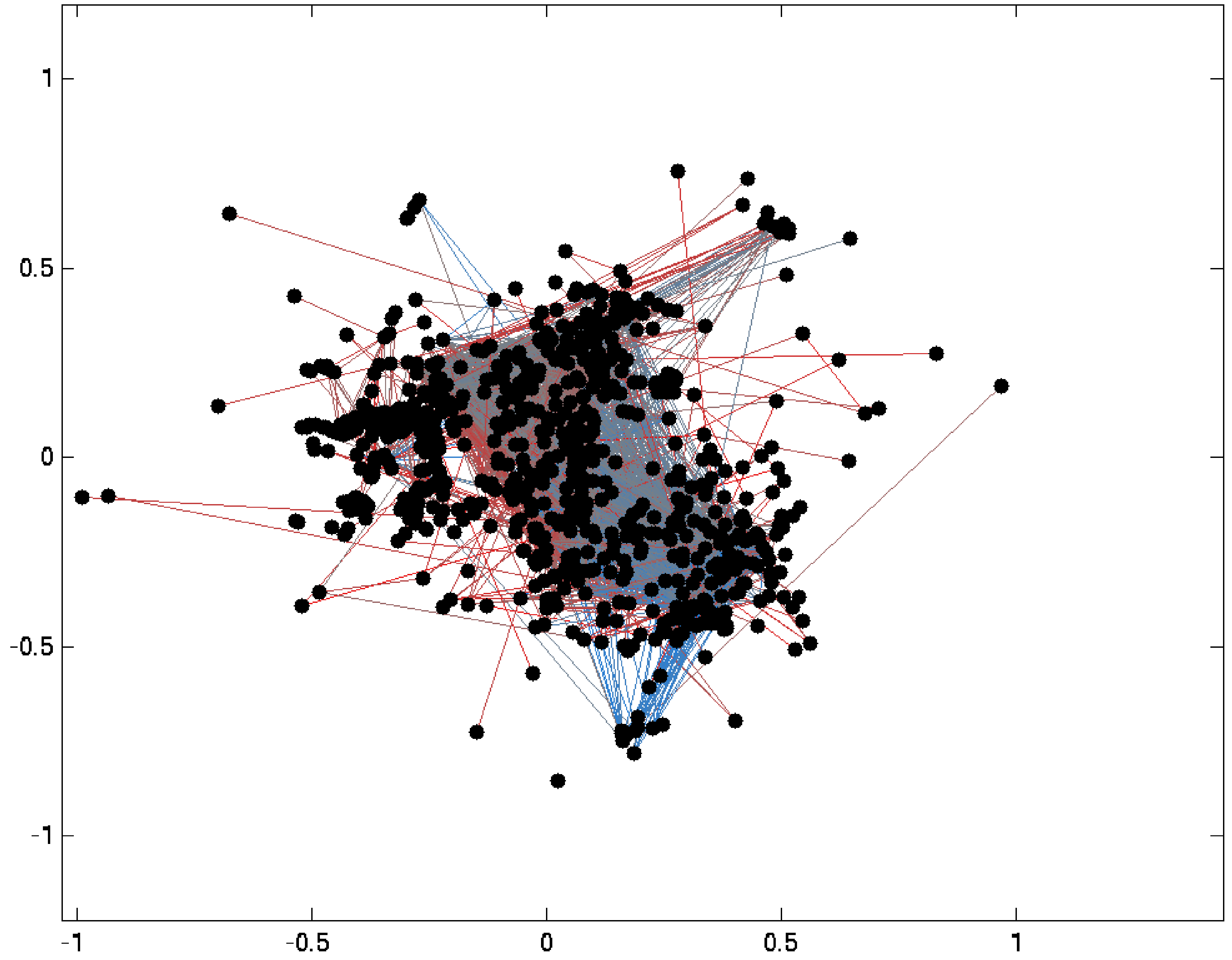


Iteratively minimizing the Kamada-Kawai cost function with weighted edges based on the distance: 30*500 iterations

US airport network

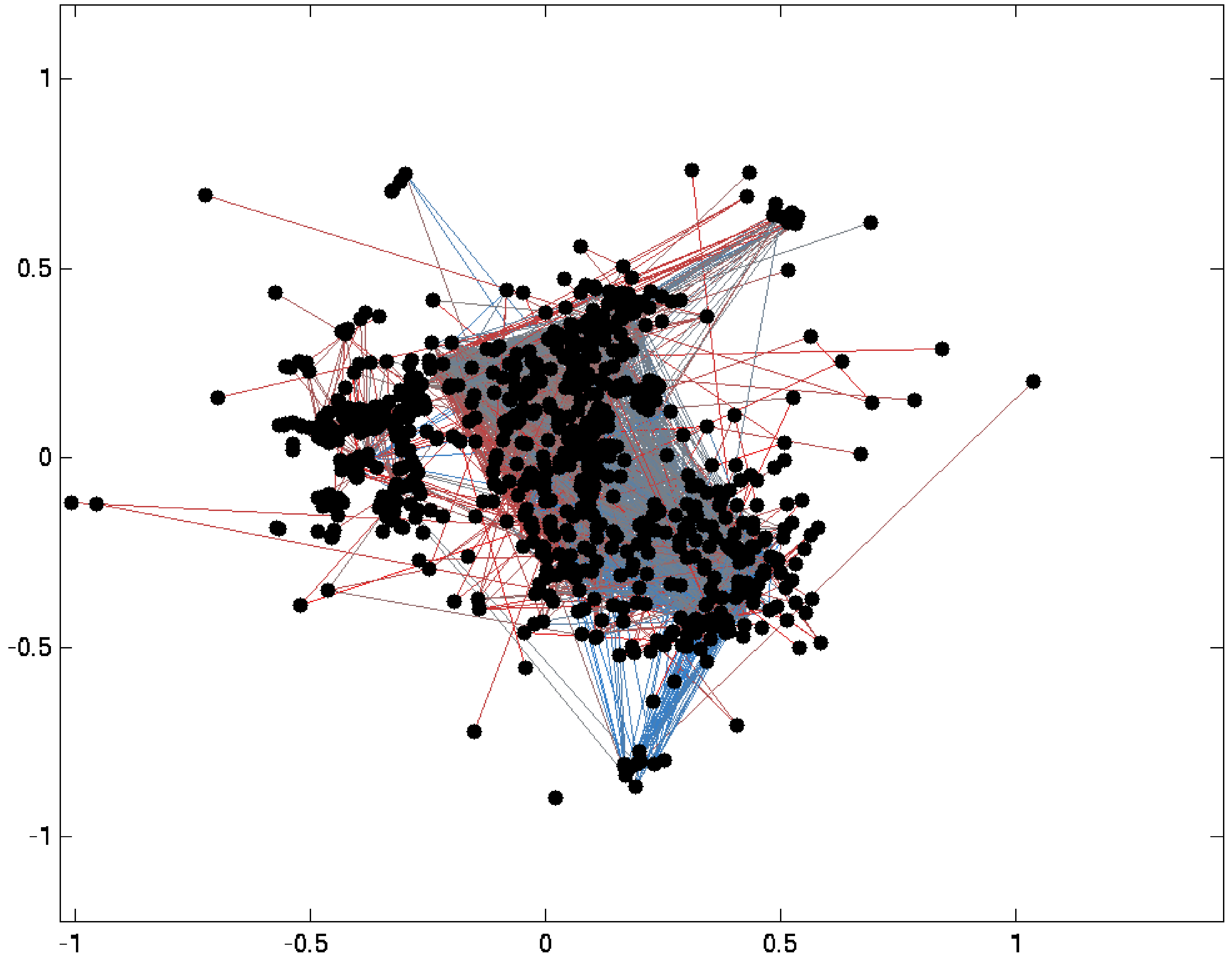
The system tries to find a balance between the "springs"

---> vertices whose edges have small desired distance tend to move in groups away from the more dissimilar vertices.



Iteratively minimizing the Kamada-Kawai cost function with weighted edges based on the distance: 35*500 iterations

US airport network



The system tries to find a balance between the “springs”

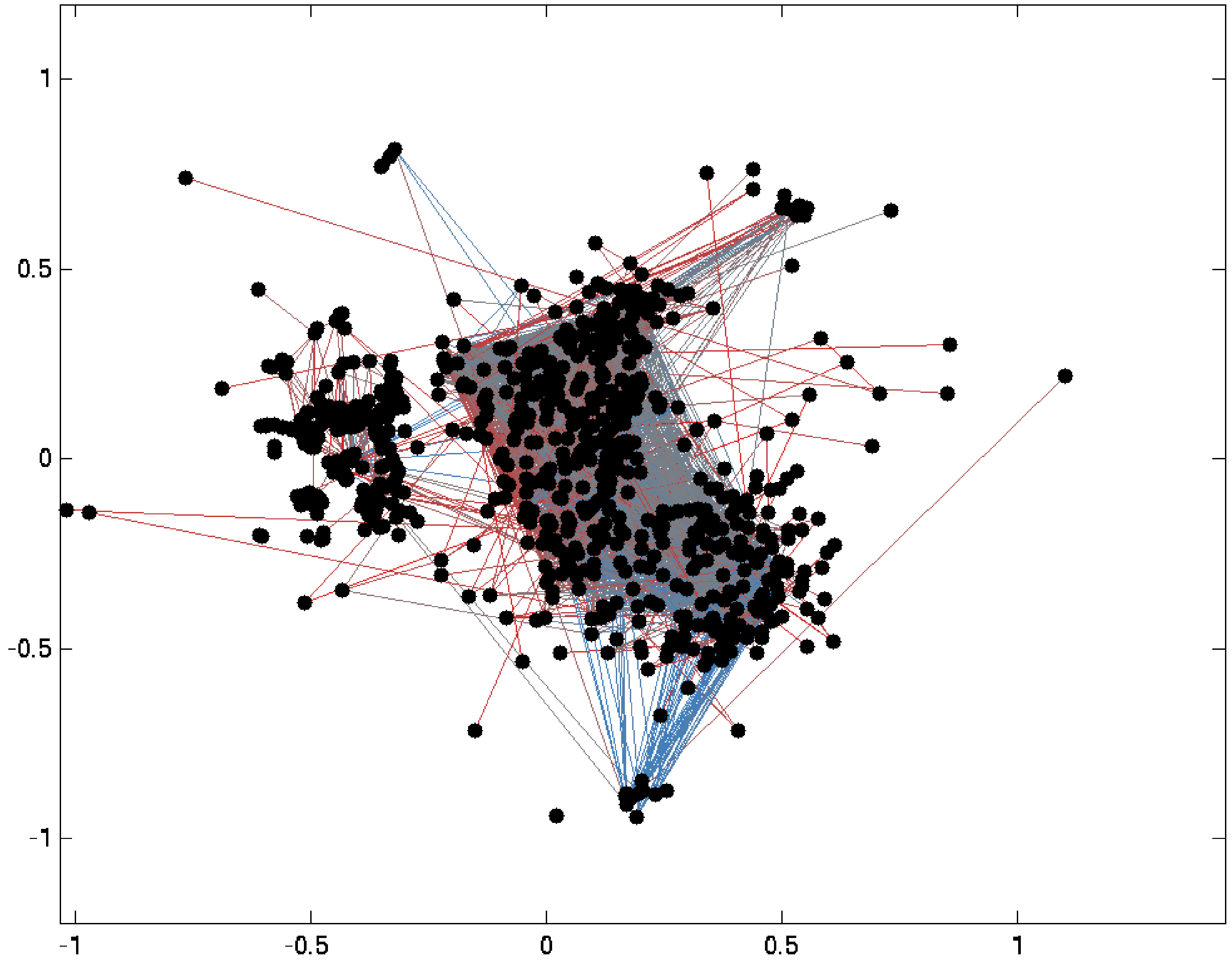
---> vertices whose edges have small desired distance tend to move in groups away from the more dissimilar vertices.

Iteratively minimizing the Kamada-Kawai cost function with weighted edges based on the distance: 40*500 iterations

US airport network

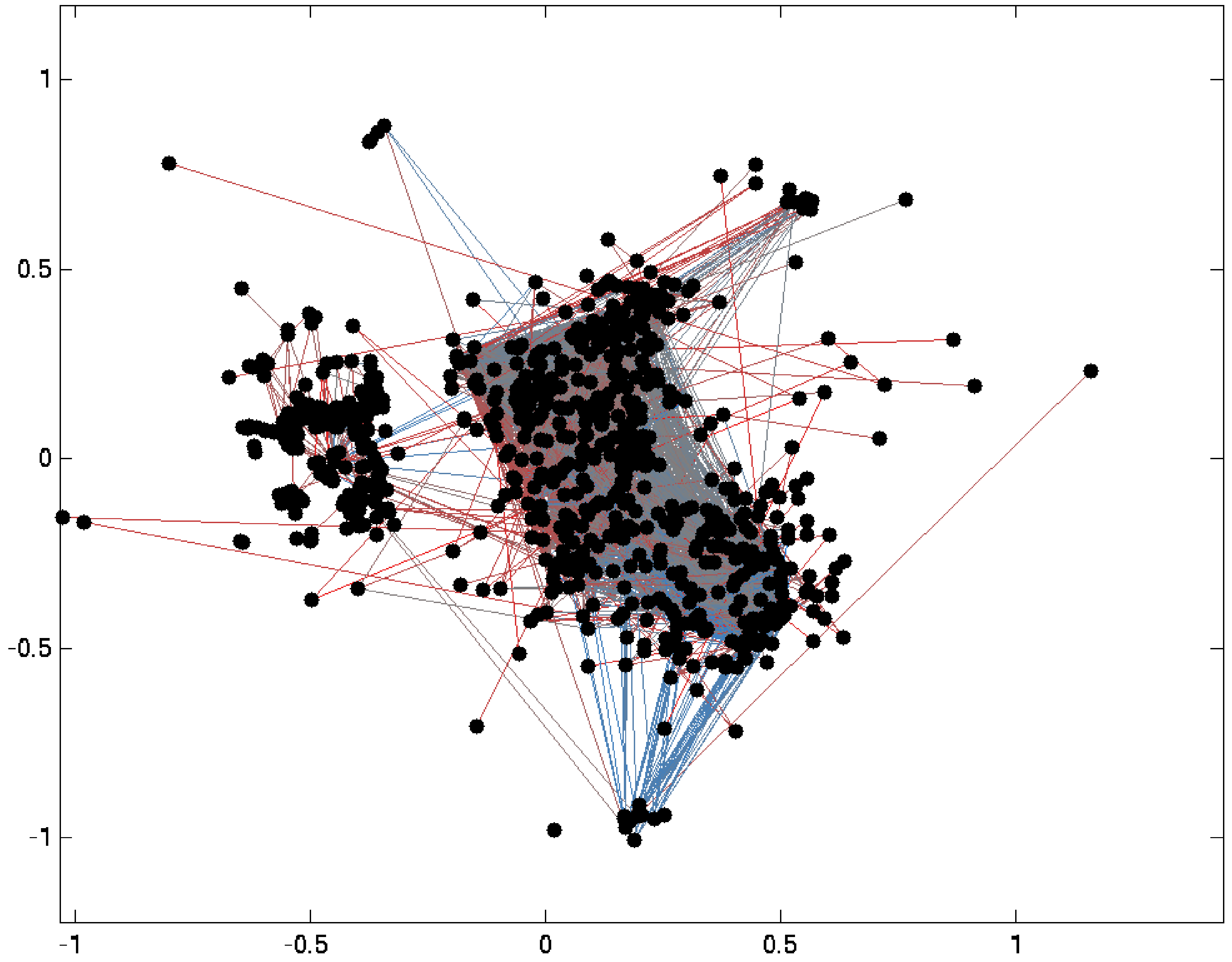
The system tries to find a balance between the "springs"

---> vertices whose edges have small desired distance tend to move in groups away from the more dissimilar vertices.



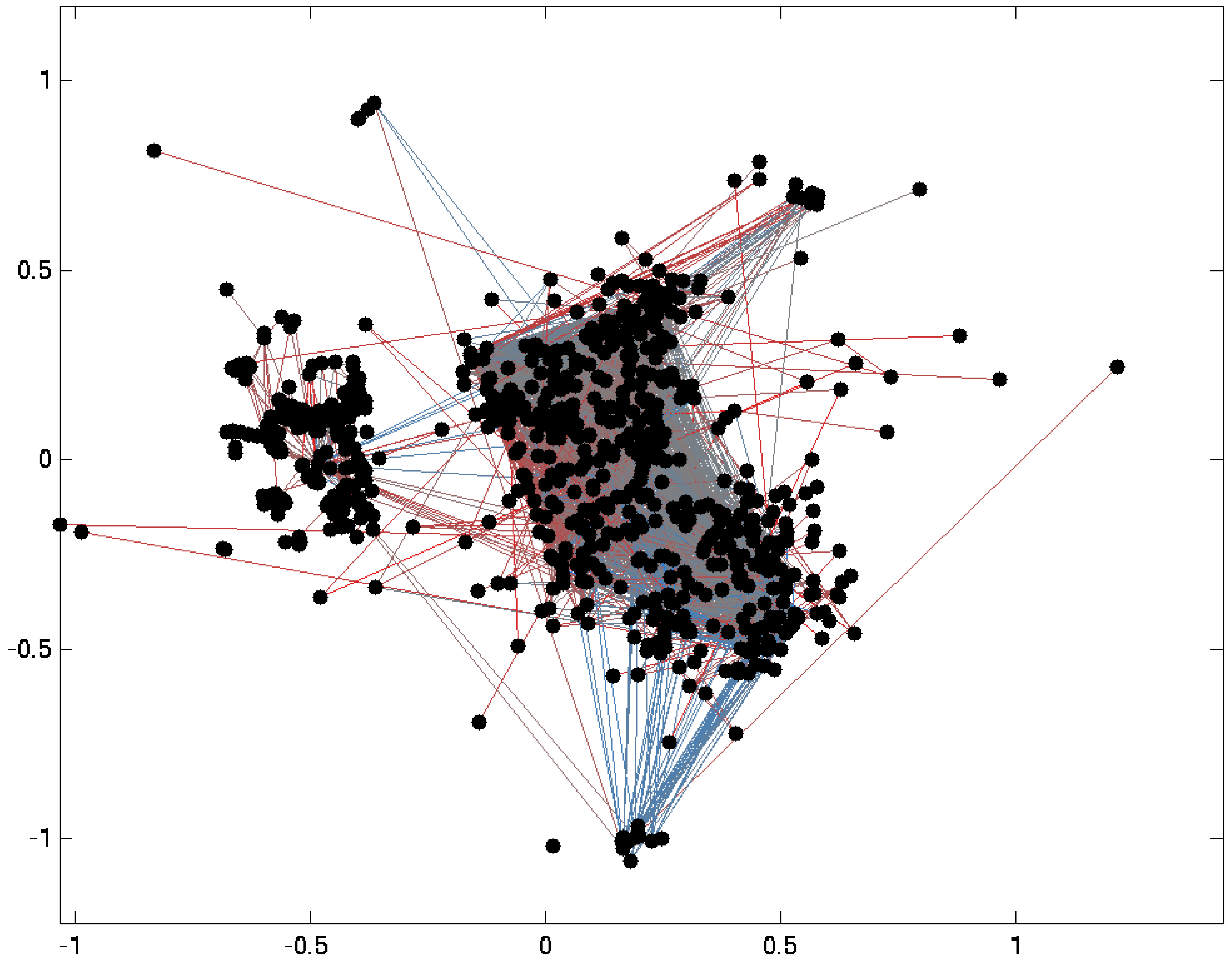
Iteratively minimizing the Kamada-Kawai cost function with weighted edges based on the distance: 45*500 iterations

US airport network



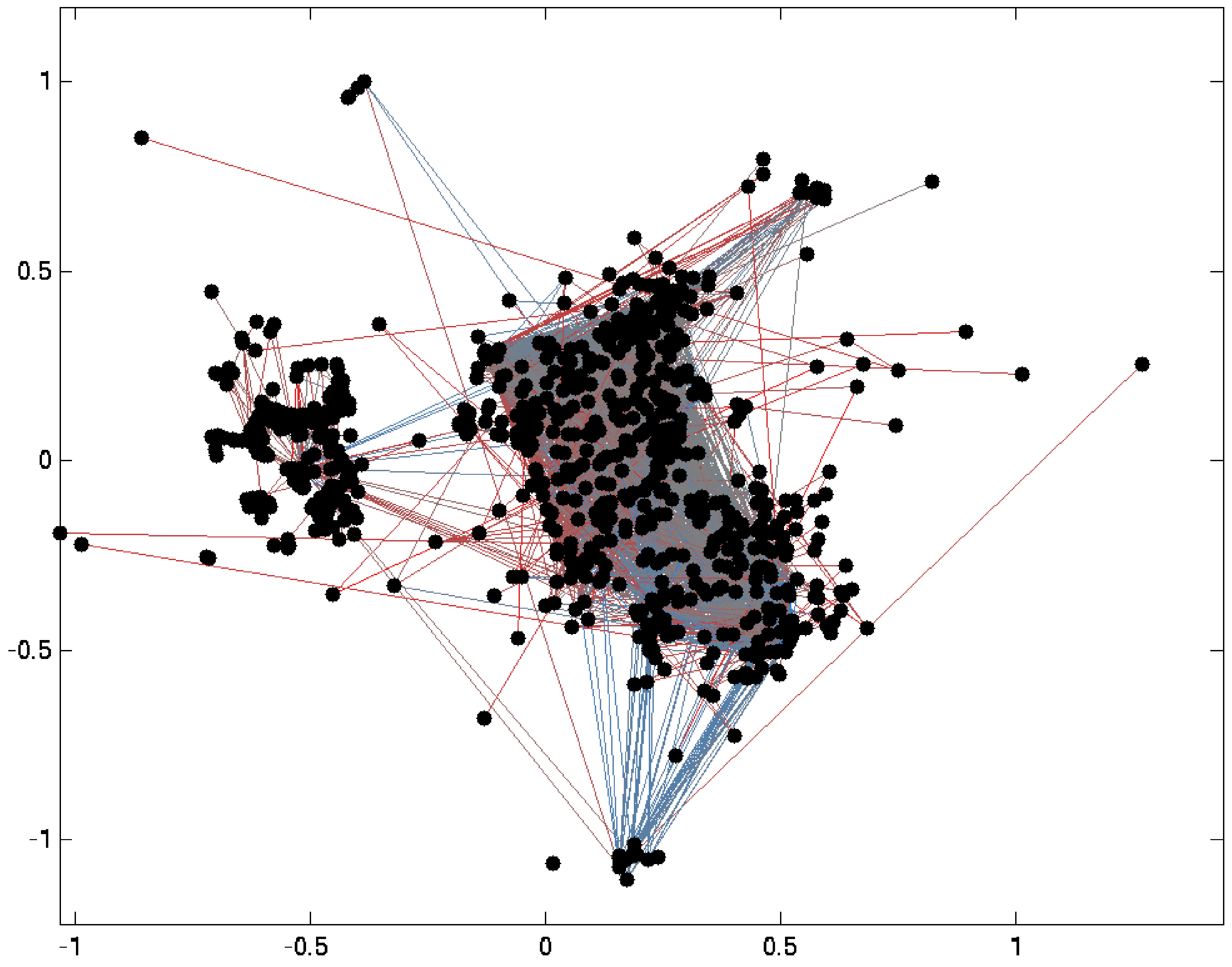
Iteratively minimizing the Kamada-Kawai cost function with weighted edges based on the distance: 50*500 iterations

US airport network



Iteratively minimizing the Kamada-Kawai cost function with weighted edges based on the distance: 55*500 iterations

US airport network

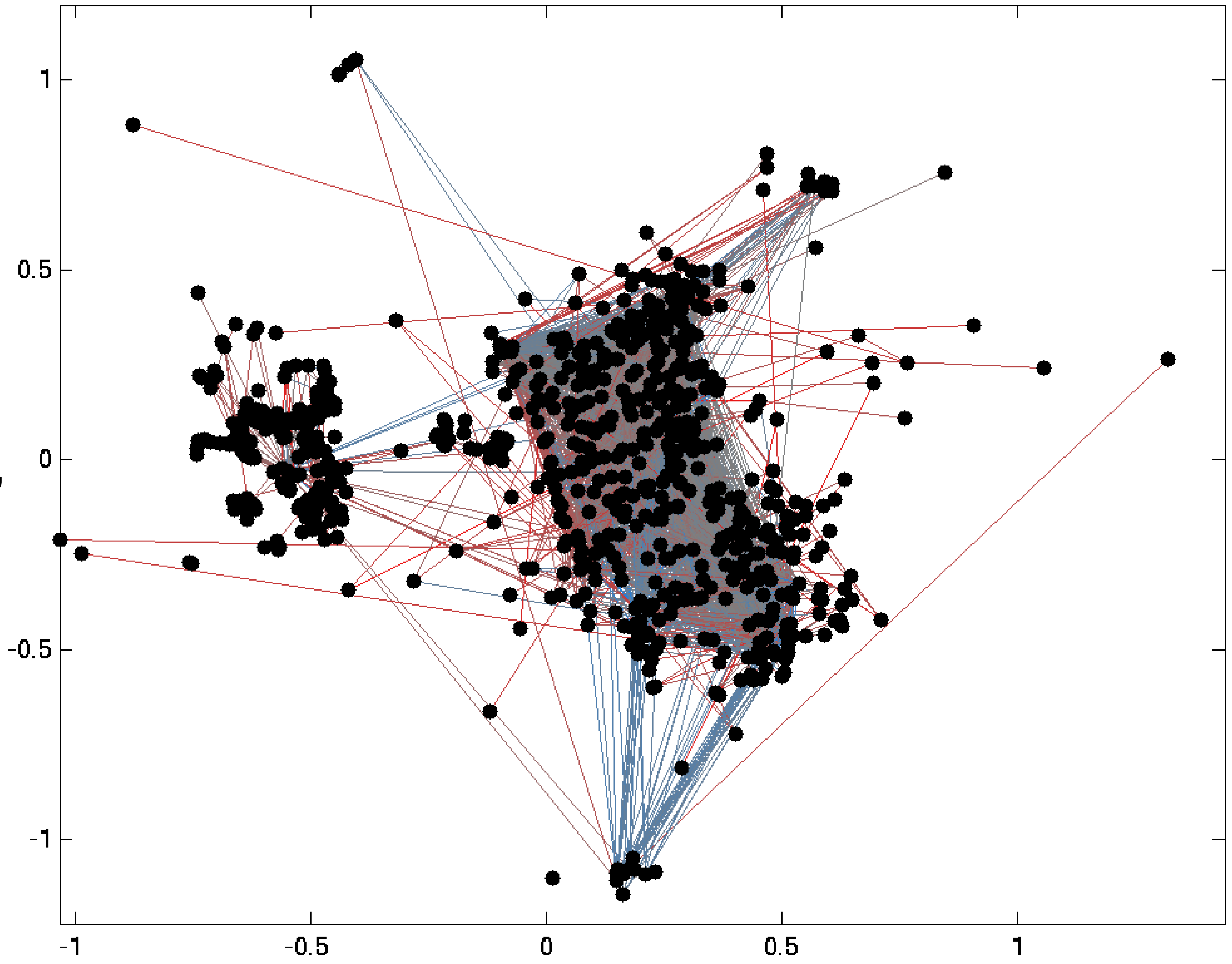


Iteratively minimizing the Kamada-Kawai cost function with weighted edges based on the distance: 60*500 iterations

US airport network

The iteration could keep going even further.

Notice how the edges are now more “grayish” -- closer to their desired lengths

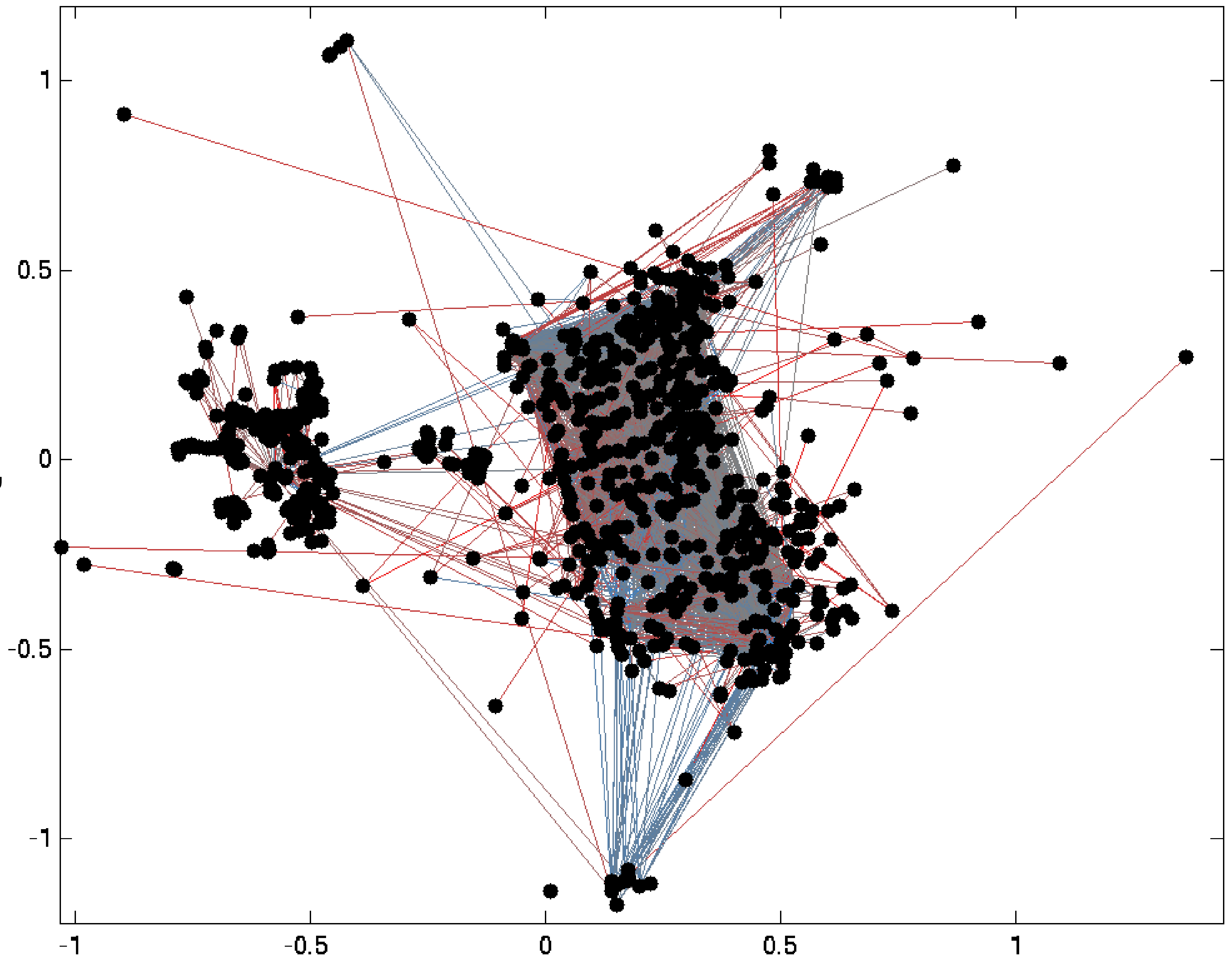


Iteratively minimizing the Kamada-Kawai cost function with weighted edges based on the distance: 65*500 iterations

US airport network

The iteration could keep going even further.

Notice how the edges are now more “grayish” -- closer to their desired lengths

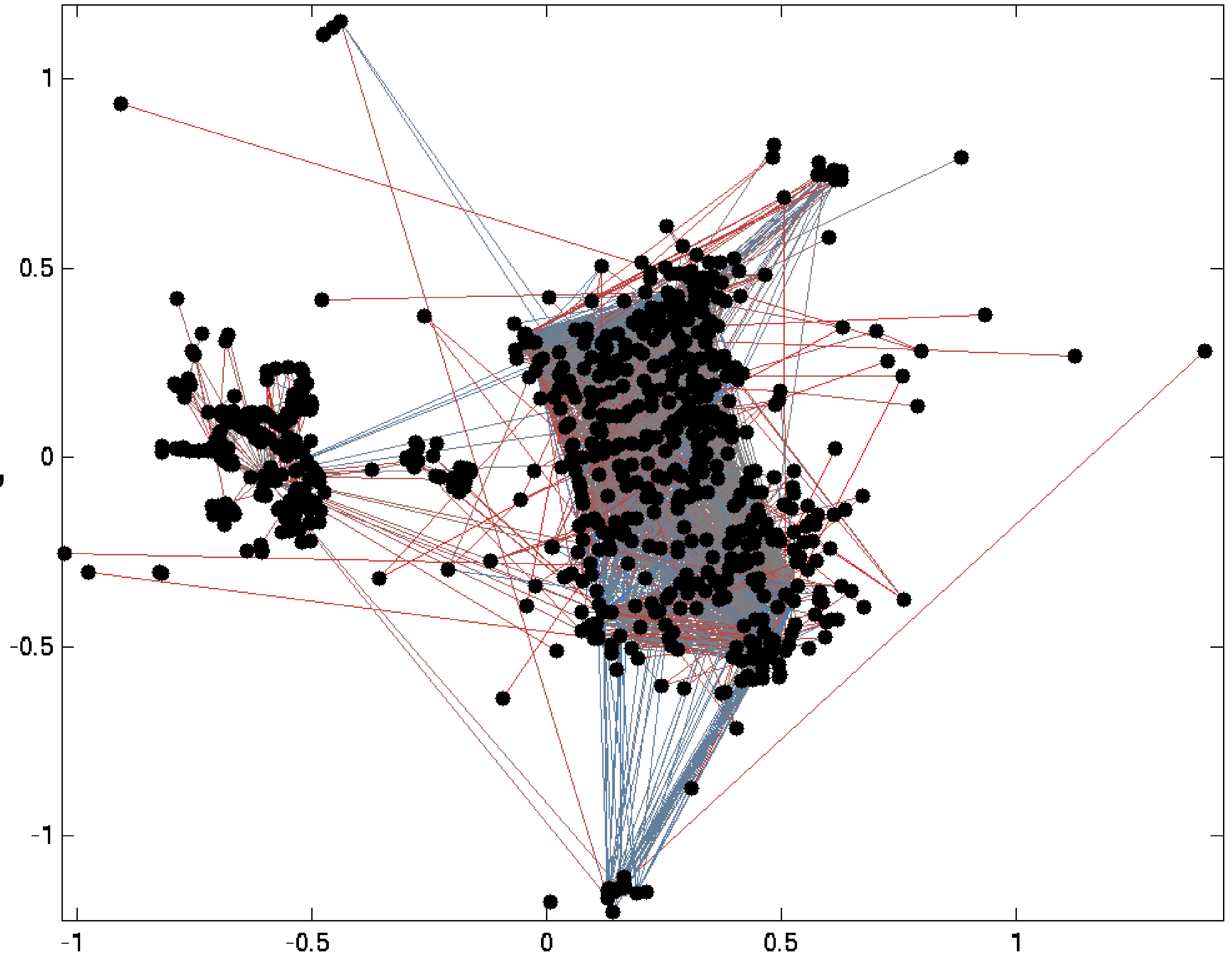


Iteratively minimizing the Kamada-Kawai cost function with weighted edges based on the distance: 70*500 iterations

US airport network

The iteration could keep going even further.

Notice how the edges are now more “grayish” -- closer to their desired lengths

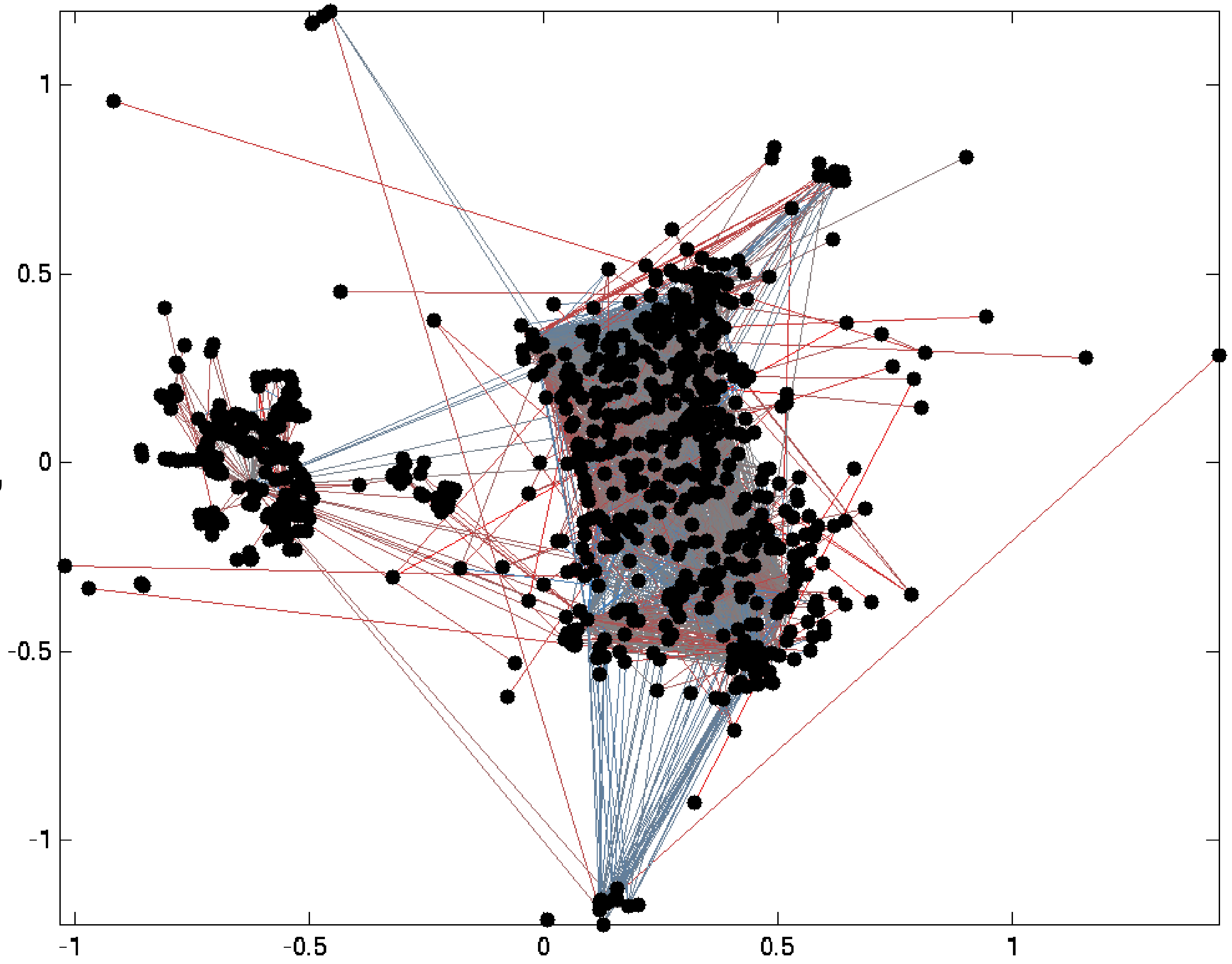


Iteratively minimizing the Kamada-Kawai cost function with weighted edges based on the distance: 75*500 iterations

US airport network

The iteration could keep going even further.

Notice how the edges are now more "grayish" -- closer to their desired lengths



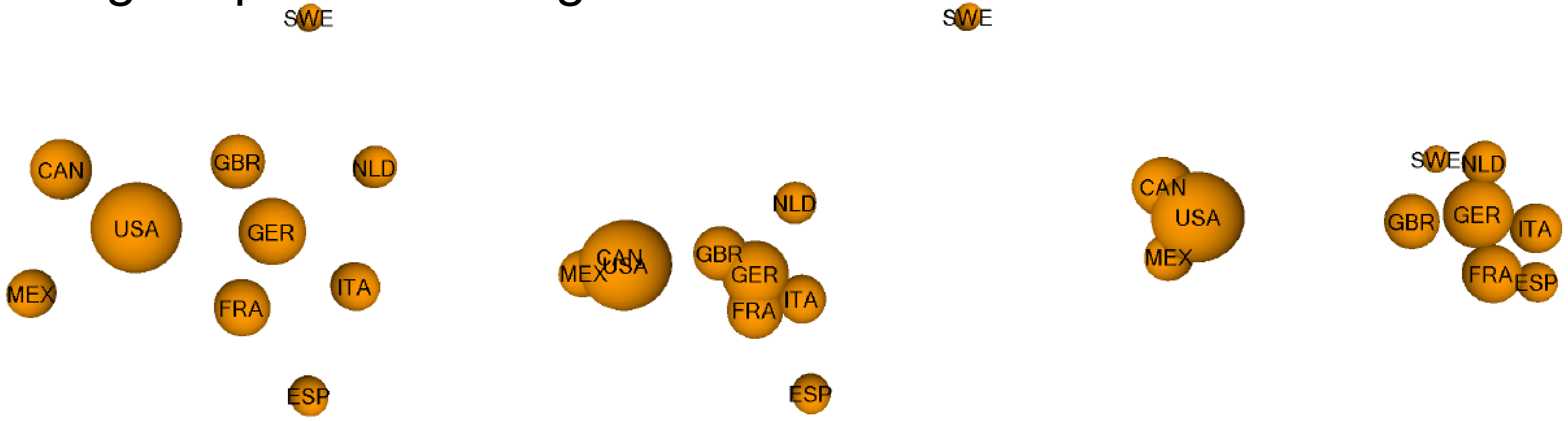
Iteratively minimizing the Kamada-Kawai cost function with weighted edges based on the distance: 80*500 iterations

Images from Andreas Noack. Energy Models for Graph Clustering. Journal of Graph Algorithms and Applications, 11(2):453-480, 2007.

LinLog

Another algorithm based on minimizing an energy function: two variants of the energy function, which represent the **cluster structure** of the graph.

The variants are called Node-repulsion LinLog and Edge-repulsion LinLog.



(a) Fruchterman-Reingold (b) Node-repulsion LinLog (c) Edge-repulsion LinLog

Comparison of LinLog variants to a classical algorithm on a graph of trade between countries. Edge-repulsion LinLog is closest to physical locations of countries.

Images from Andreas Noack. Energy Models for Graph Clustering. Journal of Graph Algorithms and Applications, 11(2):453-480, 2007.

LinLog, continued

Most energy models try to enforce small, uniform edge lengths (like the Kamada-Kawai method does). Can prevent separation of clusters, and group high-degree nodes at the center of the layout. LinLog criteria try to avoid these issues.

Node-repulsion LinLog:

$$Cost_{NodeLinLog} = \underbrace{\sum_{\{u,v\}\in E} \|\boldsymbol{y}_u - \boldsymbol{y}_v\|}_{\text{attraction between connected nodes}} - \underbrace{\sum_{u\in V,\, v\in V} \ln \|\boldsymbol{y}_u - \boldsymbol{y}_v\|}_{\text{repulsion between all nodes}}$$

It can be shown that layouts that minimize the Node-repulsion LinLog cost minimize the ratio of the **mean distance between connected nodes** to the **mean distance between all nodes**.

Images from Andreas Noack. Energy Models for Graph Clustering. Journal of Graph Algorithms and Applications, 11(2):453-480, 2007.

LinLog, continued

Edge-repulsion LinLog:

$$Cost_{EdgeLinLog} = \sum_{\{u,v\}\in E} \|\boldsymbol{y}_u - \boldsymbol{y}_v\| - \sum_{u\in V, v\in V} deg(u)\, deg(v) \ln \|\boldsymbol{y}_u - \boldsymbol{y}_v\|$$

degree of node u = number of edges connected to node u

Consider each node v as a set of deg(v) “end nodes” of edges. Then can be shown that layouts that minimize the Edge-repulsion LinLog cost minimize the ratio of the **mean distance between connected end nodes** to the **mean distance between all end nodes**.

Images from Andreas Noack. Energy Models for Graph Clustering. Journal of Graph Algorithms and Applications, 11(2):453-480, 2007.

LinLog, continued

If the graph layout is related to the graph's cluster structure, then nodes of the same dense subgraph should be close to each other, and nodes of sparsely connected subgraphs should be separated.

Minimizing distances between connected nodes, but otherwise maximizing distances between nodes, achieves this. Thus LinLog layouts try to reveal the graph's cluster structure.

Images from Andreas Noack. Energy Models for Graph Clustering. Journal of Graph Algorithms and Applications, 11(2):453-480, 2007.

LinLog, continued

It can further be shown that if two dense clusters V_1, V_2 are sparsely connected, their distance on the...

- ...Node-repulsion LinLog layout approximates their **inverse node-normalized cut** $1/ncut(V_{1,} V_2)$

$$ncut(V_{1,} V_2) = \frac{cut(V_{1,} V_2)}{|V_1| \cdot |V_2|}$$

number of edges between V_1 and V_2

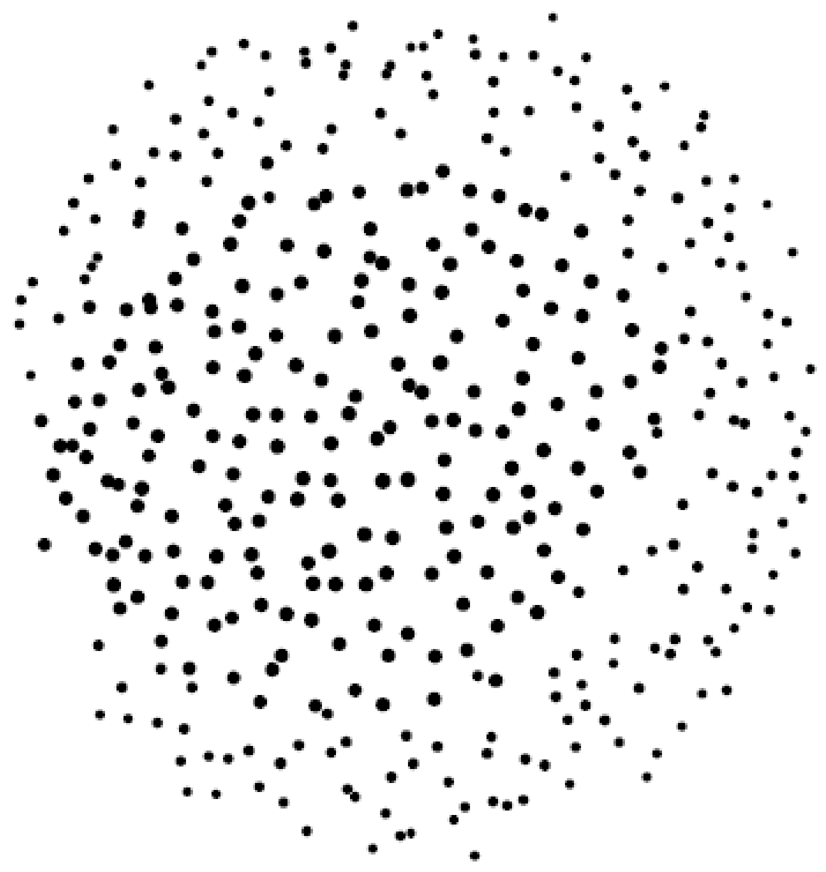
- **...**Edge-repulsion LinLog layout approximates their **inverse edge-normalized cut** $1/ecut(V_{1,} V_2)$

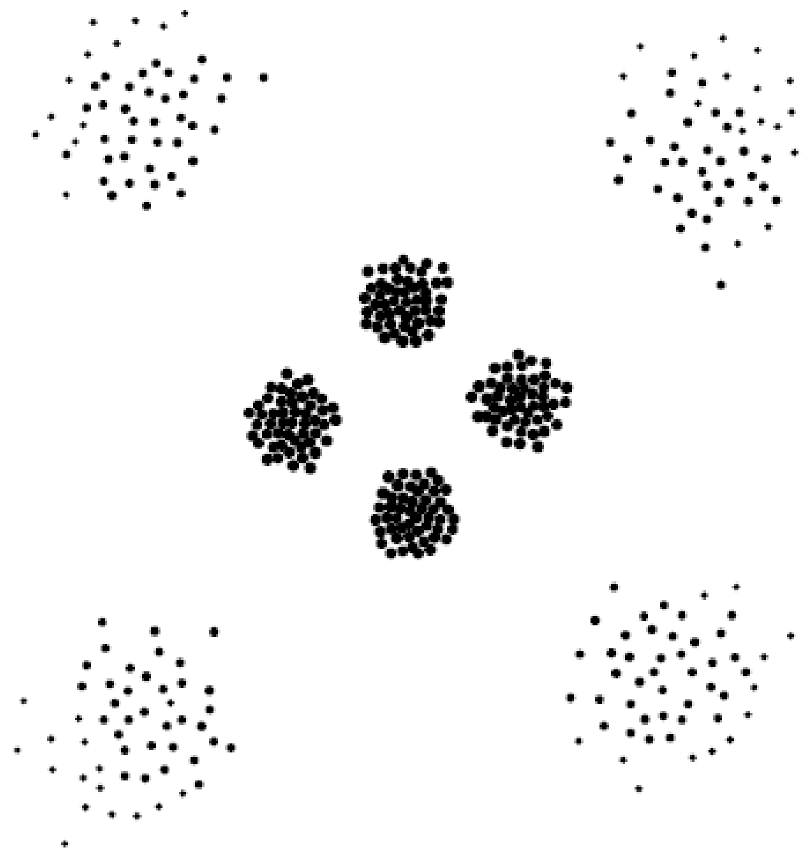
$$ecut(V_{1,} V_2) = \frac{cut(V_{1,} V_2)}{deg(V_1)\, deg(V_2)}$$

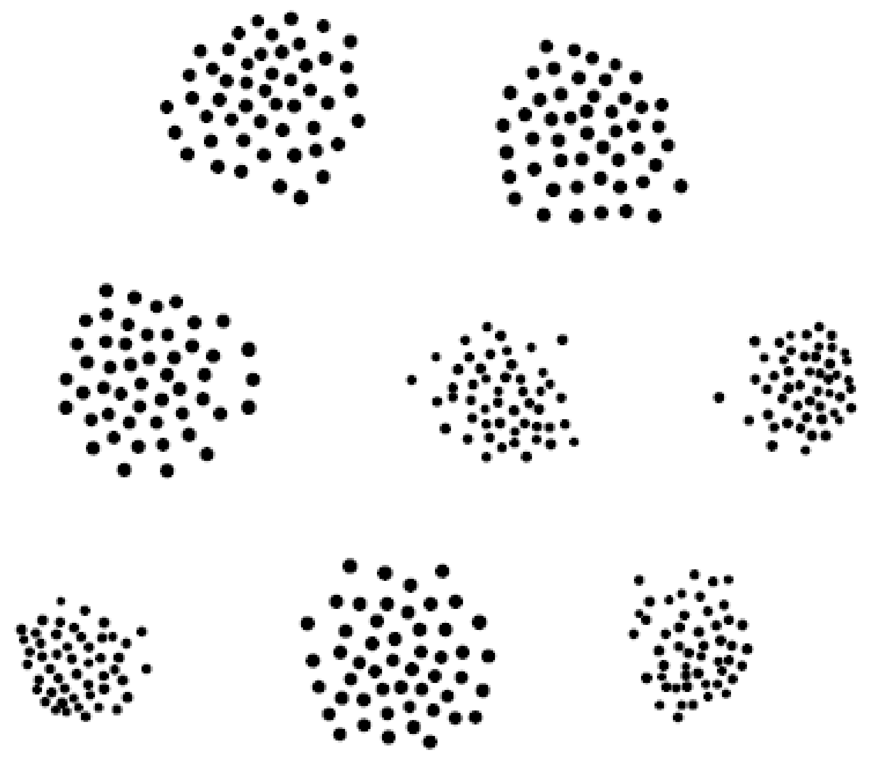
Images from Andreas Noack. Energy Models for Graph Clustering. Journal of Graph Algorithms and Applications, 11(2):453-480, 2007.

LinLog, continued

Pseudo-random graph with eight clusters in two groups. Edges sampled uniformly densely within clusters, medium-sparsely between the first group, sparsely between the second group, very sparsely between groups.


(a) Kamada-Kawai

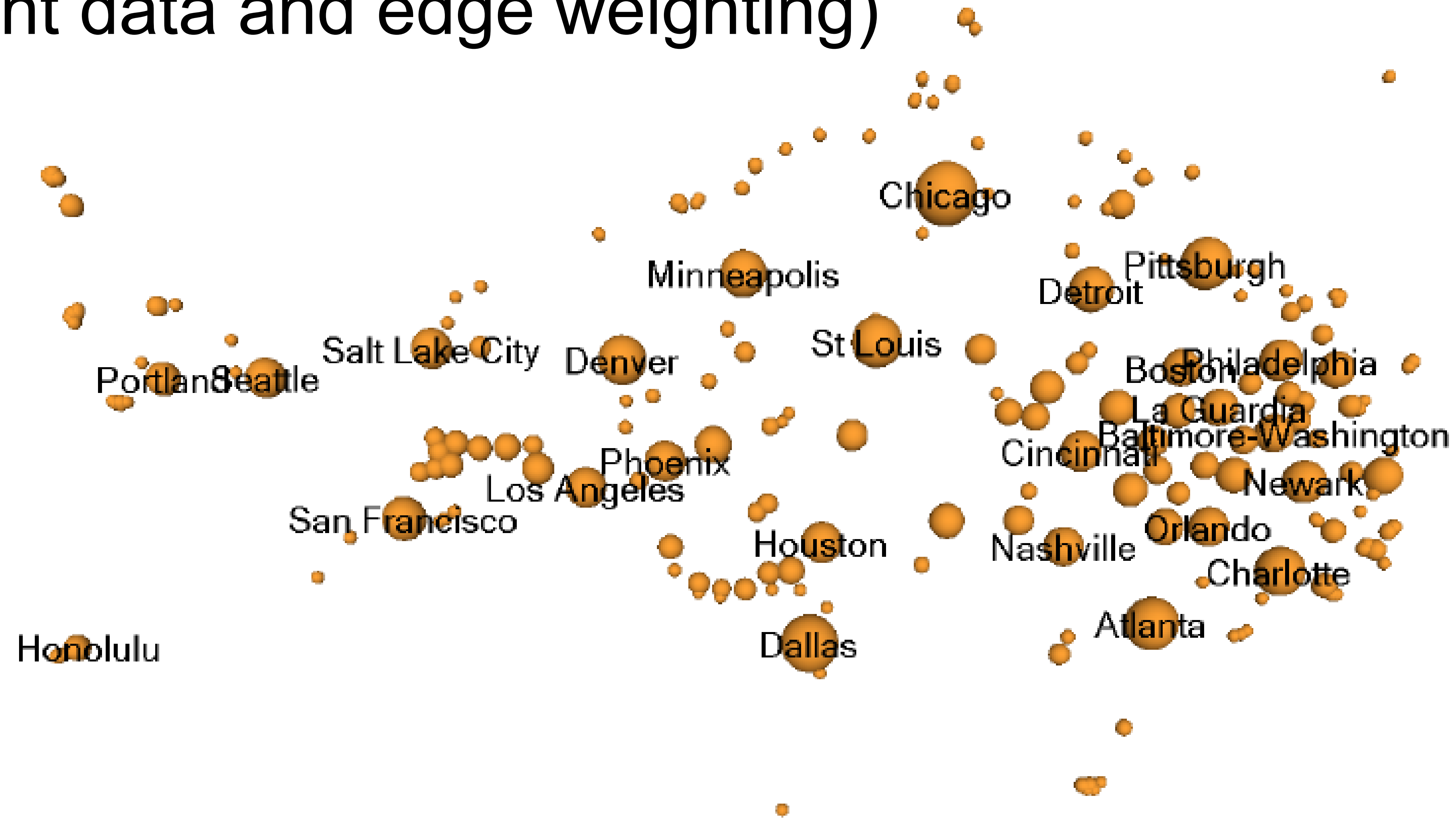

(e) Node-repulsion LinLog


(f) Edge-repulsion LinLog

Images from Andreas Noack. Energy Models for Graph Clustering. Journal of Graph Algorithms and Applications, 11(2):453-480, 2007.

LinLog, continued

Edge-repulsion Linlog on US airports data (note: may use slightly different data and edge weighting)

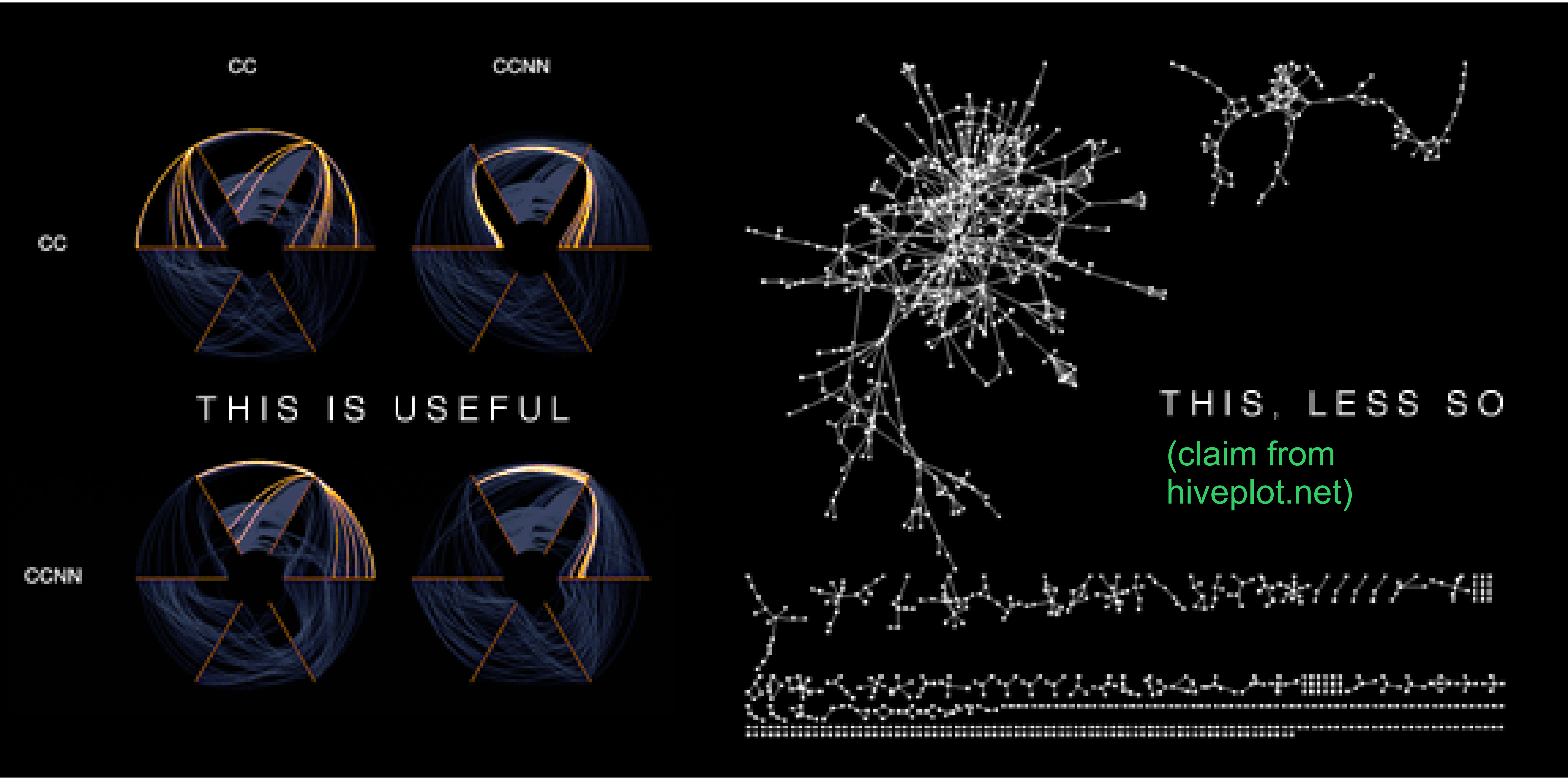


(c) Edge-repulsion LinLog model

Figure 5: Direct flights between US airports (332 nodes, 2126 edges). Some distant airports in Alaska and the South Sea (e.g. Guam) are omitted to improve readability.

Hive plots

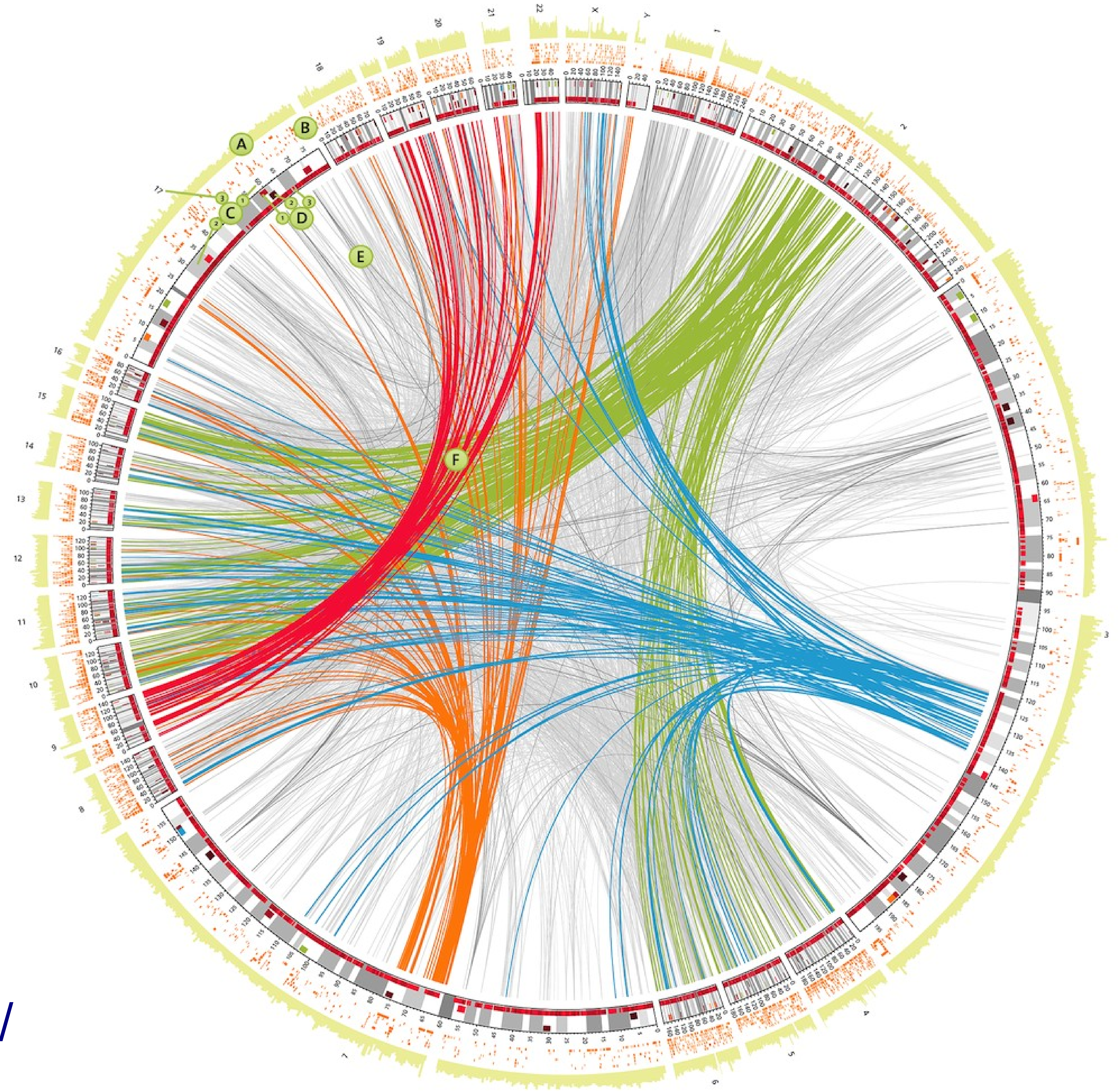
http://www.hiveplot.net/



Nodes are positioned on radially distributed linear axes, based on structural properties of the network. Edges are drawn as curved links.

Circos

computes
circular layouts



http://circos.ca//

Part 6:
Interactive visualization of graphs

Navigation and interaction

Navigation and interaction are essential in infoviz

- Layout algorithms alone cannot overcome the problems raised by the large sizes of the graphs occurring in many applications

Zoom and pan (cont.)

Zoom and pan are conventional tools in visualization

- Indispensable when large graph structures are to be explored

Zooming is well-suited for graphs, because the graphics used to display them are usually fairly simple

- Mostly lines, sometimes curves, and other simple geometric forms

Zoom and pan (cont.)

Zooming can take on two main forms

- **Geometric:** simply provides a blowup of the graph content
- **Semantic:** information content changes and more details
are shown when approaching a particular area of the graph

http://ucjeps.berkeley.edu/map2.html

Zoom and pan is thus conceptually simple, but it might create problems when used in interactive environments

Zoom and pan (cont.)

Road map of Europe, the user has zoomed into the area around London but wants to change to a view of Berlin

- Doing this without changing the zoom factor (at least temporarily) might be slow (zoom out, pan to Berlin and zoom in again)
- The user wants smooth changes (perform zoom and pan in parallel)

When zooming in, the view expands exponentially fast and the target point moves always faster than the pan can keep up with

The net result is a target that is approached non-monotonically

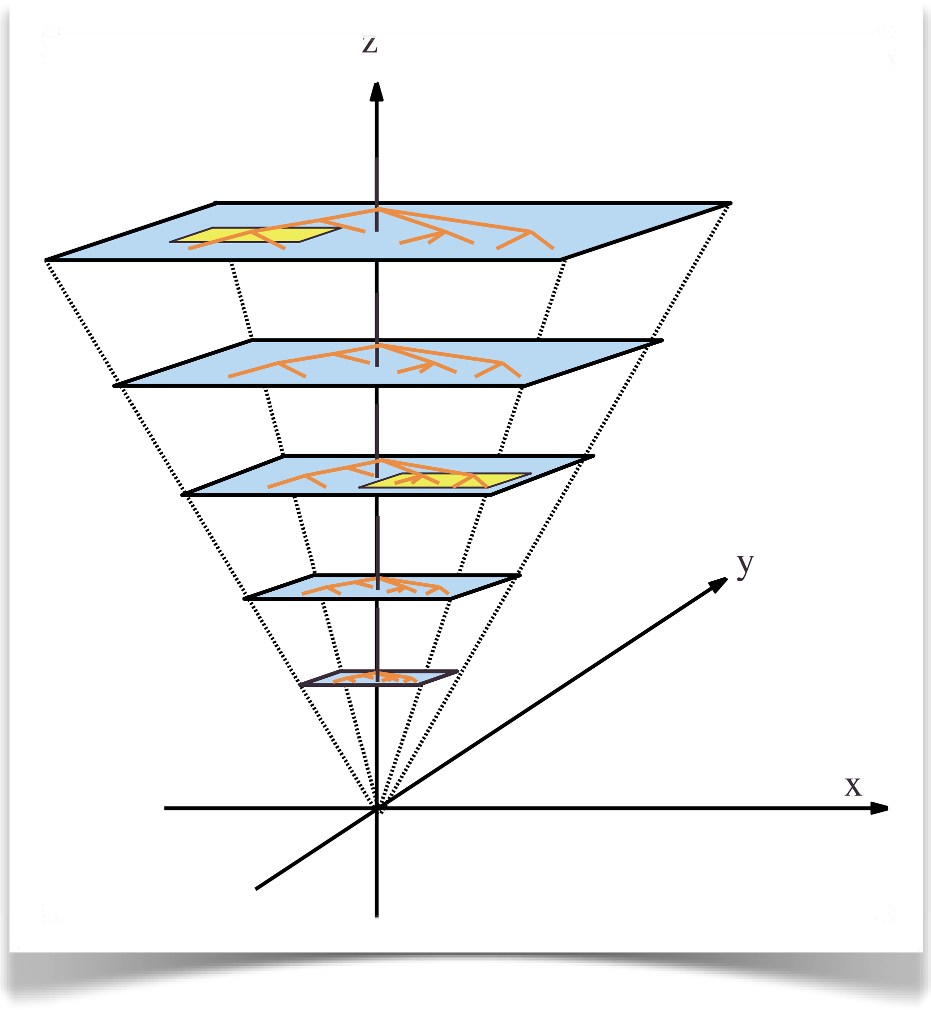
Zoom and pan (cont.)

Furnas and Bederson proposed an elegant solution to alleviate the effects of the zoom and pan problem

Space-scale diagrams

Create copies of the original 2D picture, one at each possible magnification and stack them up

- Combinations of (continuous) zoom and pan actions are described as paths
- An optimal path in this abstract space can be found e.g., thru minimum path length



Focus and context

Another well-known problem with zooming is that if one zooms on a focus, all contextual information is lost

- The loss of context can become a considerable usability obstacle

A set of approaches allow to focus on some detail without losing the context

Focus+Context techniques

They do not replace zoom and pan but rather complement them

Focus and context (cont.)

Graphical fish-eye distortion is a popular technique for F+C

- Enlarging the area of interest and showing other portions of the image with successively less detail

Conceptually, the graph is mapped onto the plane and a "focus" point is defined by the user

- The distance from the focus to each node of the tree is then distorted by a function and the distorted points and connecting edges are displayed

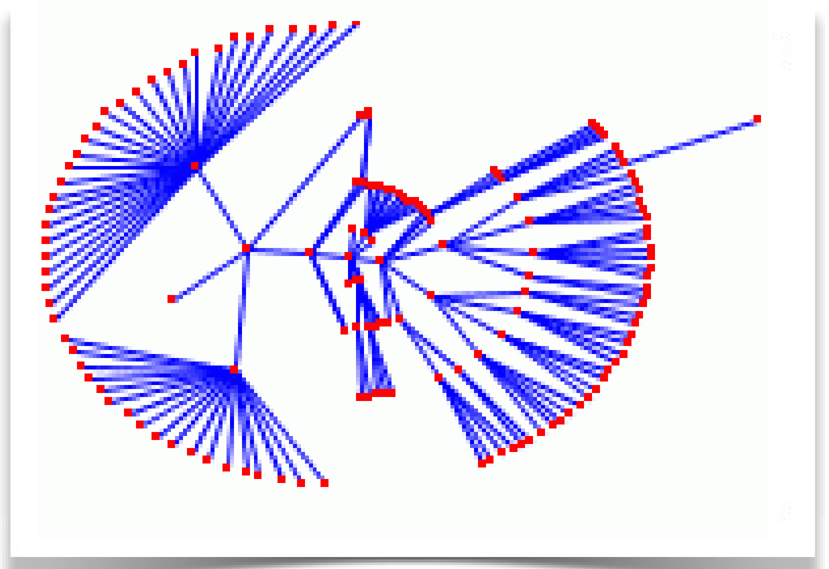
Focus and context (cont.)

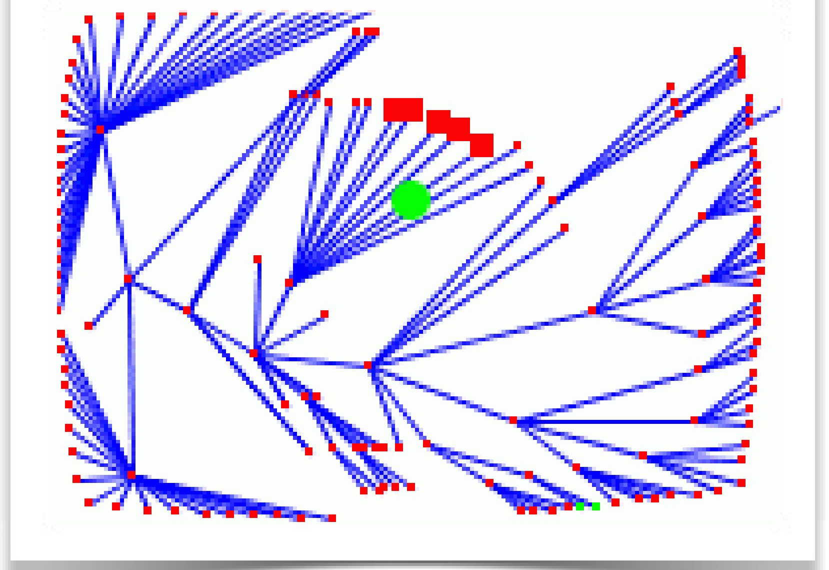
The created distortion depends on the form of the distorting function

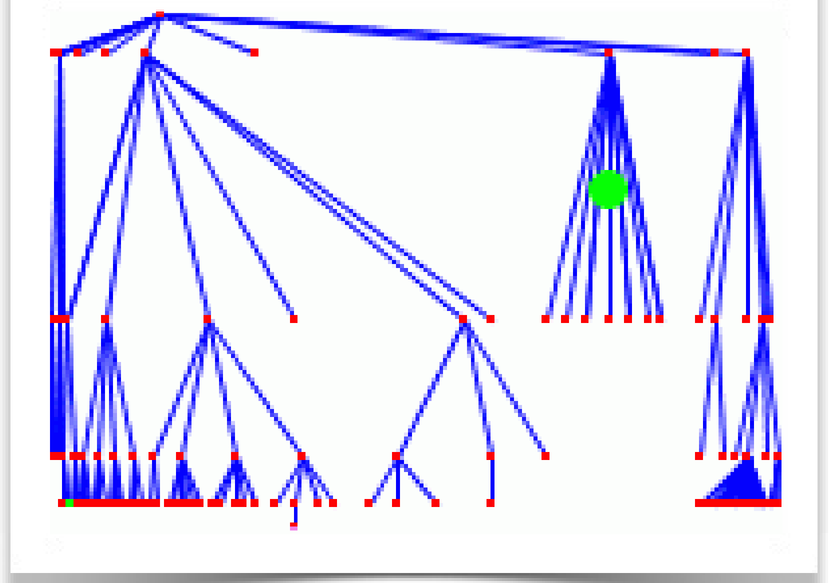
Two basic variations:

- Polar distortion: It applies to the nodes radially in all directions, starting from the focus point

- Cartesian distortion: It is applied on each direction (x and y) independently before establishing the final position of the nodes







Focus and context (cont.)

This simple but powerful technique is an important form of navigation but has at least one major pitfall

- The essence of the fish-eye is to distort the position of each node
- If the distortion is faithfully applied, straight edges connecting the nodes will also be distorted, the result is a general curve

Standard graphic systems don't seem to offer the necessary facilities to transform lines into curves

- Mostly because of the prohibitively large number of calculations
- ... so, we get straight-line edges and distortion only for nodes

The consequence of this solution is edge-crossing

Focus and context (cont.)

Interaction with fish-eye means changing the position of the focus and/or modifying the distortion parameters

The fish-eye technique is independent of the layout algorithm and it is defined as a separate postprocessing step on the graphical layout of the graph

- **(+)** modular organization of the software implementation
- **(-)** may destroy the aesthetics governing the layout algorithm

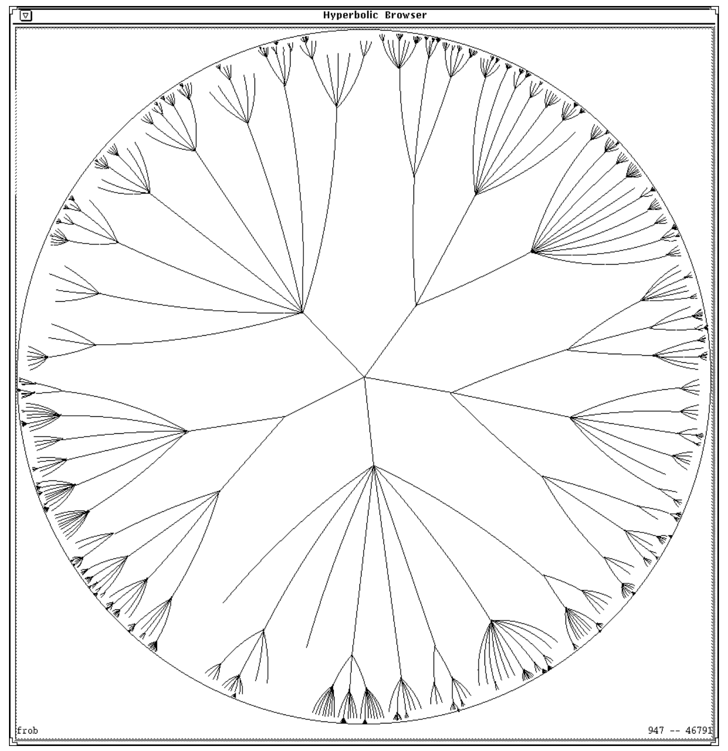
Appropriate distortion possibilities can be built into the layout algorithm itself

- Context+Focus effects that merge with Layout

Focus and context (cont.)

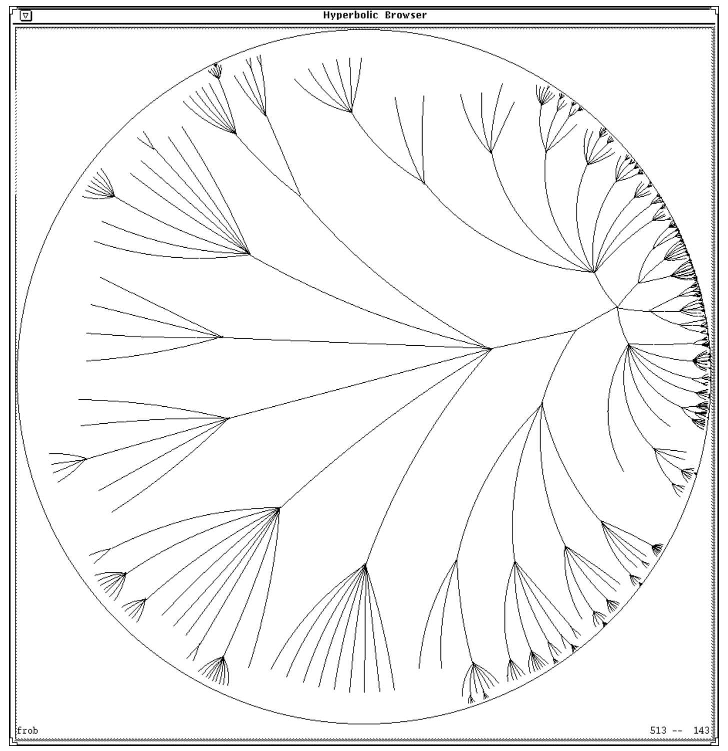
The hyperbolic layout does just that

- Whether in 2D or 3D, it produces a distorted view with a focal point at some fixed location in the graph, in a fish-eye view sense



However, proper interaction with the view means changing the position of the center point within the graph

- The root has been shifted to the right, putting more focus on the nodes that were toward the left



Software

- **GraphViz**: http://www.graphviz.org - open source visualization software, includes implementation of some layout optimization algorithms.
- **Cytoscape**: http://www.cytoscape.org - open source, oriented for bioinformatics data, includes some layout optimization algorithms
- **Gephi**: http://gephi.org - open source, includes some layout optimization algorithms

Other resources:

- www.graphdrawing.org
- graph drawing tutorial: http://cs.brown.edu/people/rt/papers/gd-tutorial/gd-contraints.pdf
- Journal of Graph Algorithms and Applications: open access journal, http://jgaa.info

References

I. Herman, G. Melançon and S. Marshall, “Graph visualization and navigation in information visualization: A survey”, *IEEE Transactions on Visualization and Computer Graphics*, 6(1), 24-43, 2000

Manuel Lima, *Visual complexity: Mapping patterns of information*. Princeton Architectural Press (2011)

Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. Algorithms for drawing graphs: An annotated bibliography. Computational Geometry: Theory and Applications, 4(5):235{282, 1994. http://dx.doi.org/10.1016/0925-7721(94)00014-X.

David Easley and Jon Kleinberg. Networks, Crowds, and Markets: Reasoning about a Highly Connected World. Cambridge University Press, 2010. http://www.cs.cornell.edu/home/kleinber/networks-book/.

References, continued

Michael Hahsler, Kurt Hornik, and Christian Buchta. Getting things in order: An introduction to the r package seriation. Journal of Statistical Software, 25(3):1{34, 2008. URL http://www.jstatsoft.org/v25/i03.

Robert Kosara. Graphs beyond the hairball, 2012. URL http://eagereyes.org/techniques/graphs-hairball. Blog entry.

Carlos E. Scheidegger. So you want to look at a graph, part 1, 2012. URL http://cscheid.net/blog/so_you_want_to_look_at_a_graph__part_1. Blog entry.

Roberto Tamassia, editor. Handbook of Graph Drawing and Visualization. CRC Press, 2013. http://cs.brown.edu/~rt/gdhandbook/.

Andreas Noack. Energy Models for Graph Clustering. Journal of Graph Algorithms and Applications, 11(2):453-480, 2007. http://jgaa.info/accepted/2007/Noack2007.11.2.pdf

References, continued

Tomihisa Kamada and Satoru Kawai. An Algorithm for Drawing General Undirected Graphs. Information Processing Letters 31:7-15, 1989.

Andreas Noack. Energy Models for Graph Clustering. Journal of Graph Algorithms and Applications, 11(2):453-480, 2007. http://jgaa.info/accepted/2007/Noack2007.11.2.pdf