

Visualization of Document Co-Citation Counts

Steven Noel

Center for Secure Information Systems
George Mason University
Email: snoel@gmu.edu

Cheehung Henry Chu, Vijay Raghavan

Center for Advanced Computer Studies
The University of Louisiana at Lafayette
Email: {cice, raghavan}@cacs.louisiana.edu

Abstract

Visualization can facilitate the understanding of the structures of a collection of documents that are related to each other by links, such as citations in formal publications. We present results of visualizing minimum spanning trees based on document co-citation counts and on document citation correlations.

Keywords: Visualization, citation analysis, minimum spanning tree.

1. Introduction

The importance of clustering in information retrieval is well known [1]. Moreover, complex clustering results are often more easily understood through visualizations. While innovative information retrieval systems often employ link analysis for hypertext, the analysis output is usually presented as a simple linear list, with the only relationship among documents being query relevance. Useful query-independent analyses like clustering are usually either not done or the analyses are not enhanced with visualizations.

Hyperlink systems, e.g. the World Wide Web or science citations, can in general be modeled as directed graphs. A co-citation between two documents is the citing (or hypertext linking) of the two documents by another one. A measure of similarity between a pair of documents can be defined as the number of documents that co-cite the pair. This is known as citation count. Taken over all pairs of documents, the co-citation count similarity serves as a compact representation of the citation graph structure.

The minimum spanning tree (MST) has previously been proposed for visualizing document collections,

with the distances between documents computed from co-citations [2, 3]. The tree shows the minimal set of essential links among documents, which is interpreted as the network of direct influences within the collection. This provides useful analysis for information retrieval. Branches in the tree correspond to bifurcations of ideas in the evolution of science, with highly influential documents appearing near the center of the network, and the emerging research front represented as documents on the fringes.

Section 2 describes an algorithm for placing minimum spanning tree vertices in the plane, so that the tree can be visualized. In Section 3, we present our results of using MST based on co-citation counts and on correlation. We draw our concluding remarks and suggest future work in Section 4.

2. Placement of MST vertices

A minimum spanning tree for an undirected graph is the minimum-distance tree that connects all vertices in the graph. Thus vertices are connected to one another by the smallest possible distance between them. In terms of hypertext systems with distances based on co-citation, a minimum spanning tree edge is interpreted as the most direct influence between two documents. The most well known algorithms for computing the minimum spanning tree are Kruskal's algorithm and Prim's algorithm. Both can be easily implemented in $O(E \log V)$ time.

While the computation of the minimum spanning tree is straightforward, it remains to put the tree in a form appropriate for visualization. The resulting visualized tree has the interpretation as a network of influences among the hyperlinked documents. Spatial coordinates must be induced for the minimum spanning tree vertices so that visualization algorithms can be applied. That is, we must spatialize this inherently non-spatial object. The only information at our disposal is the topology of the minimum spanning tree graph, and the corresponding edge weights.

A classical method for visualizing distance data is multidimensional scaling [4]. The distances are often generated from high-dimensional data, so that multidimensional scaling is seen as a method of dimensionality reduction. That is, it is a method of projecting high dimensional data into lower dimensions, while trying to preserve distances among the data items. In the case of our co-citations, document distances are computed from of high-dimensional inner products of the citation adjacency matrix columns (generalized inner products for higher-order co-citations).

Multidimensional scaling is seen as a nonlinear competitor to principal component analysis. The classical form of multidimensional scaling with Euclidean distances is equivalent to the first k principal components, where k is the dimensionality of the visualization. Like principal component analysis, multidimensional scaling attempts to compute linear combinations of the original dimensions that maximize data variance. Thus it blends together those dimensions that seem to be well correlated. Standard algorithms for multidimensional scaling are slow, e.g. $O(n^3)$ or $O(n^4)$. Also, most algorithms are non-iterative, that is, they must be started from the beginning each time, which can be a disadvantage for many applications.

Multidimensional scaling attempts to map data objects $i = 1, 2, \dots, n$ to spatial points $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ (e.g. in the plane) such that the original distances are approximately maintained. A *stress function* measures the fitness of the approximation, i.e. the fitness between the original distances and the distances for the mapped points. The simplest stress function $f(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$ is the sum of the squares of the residual, or

$$f(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n) = \sqrt{\sum_{j,k} (d_{j,k} - \|\mathbf{x}_j - \mathbf{x}_k\|)^2},$$

where $\|\bullet\|$ is the vector norm, usually Euclidean. Minimization of the stress function usually employs some form of gradient descent, though simulated annealing is sometimes applied to avoid getting stuck in local minima.

A serious problem occurs for multidimensional scaling when the original distances are not Euclidean, since there is no way to preserve the distances in the projection to lower dimensions. We illustrate this by applying multidimensional scaling to data sets of

different sizes. We generated data sets by querying the Institute for Scientific Information's Science Citation Index (SCI) using the keyword "microtubules" for publications from the Year 1999. We selected the first 100 documents returned from the query. These 100 documents cite a total of 3070 unique documents. We then retain only those cited documents that have been cited 6 or more times (this is a typical filtering in citation analysis, for retaining only the more frequently cited documents). This results in a collection of 45 documents. Our results are shown in Figure 1. Stress for only three documents is very low, since three points whose distances obey the triangle inequality can always be drawn in the plane. Stress increases dramatically with the addition of a fourth point, and continues to increase as points are added. The minimum spanning tree visualization for 45 documents is quite poor, containing many confusing crossed edges.

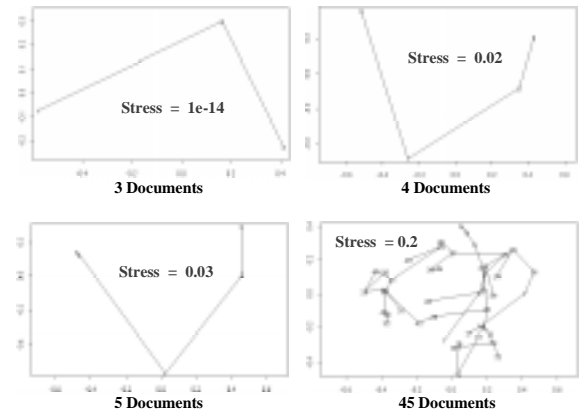


Figure 1. Multidimensional scaling for document collections of different sizes.

Multidimensional scaling is one of many optimization-based algorithms for dimensionality reduction or graph node placement. Given an error metric for node placement, these attempt to place nodes so as to minimize the metric. A frequently applied class of algorithms models a system of forces and mechanical springs in placing nodes, and was developed initially for VLSI and popularized by the work of Eades [5].

In so-called spring or force-displacement models, springs corresponding to graph edges generate forces that move vertices to better match the springs' characteristic lengths. This in turn reduces the spring-length dependent stress, interpreted as total system energy. A direct implementation of such a model is a

form of the well-known N -body simulation from celestial and atomic physics. It has time complexity $\Theta(N^2)$, since each object interacts with every other one.

But classical spring models still attempt to minimize distance errors among all pairs of items, which performs poorly for significant deviations from Euclidean distances. We employ a type of spring model that abandons the idea of minimizing distance error over all pairs [6]. That is, it makes no explicit attempt to minimize a stress formula.

Designed for general undirected graphs and not just fully connected ones, it models attractive spring forces only for edges actually in the graph. These forces cause adjacent vertices to be drawn towards one another. There are repulsive forces among all vertices, which prevent them from collapsing onto one another, and distribute them in the plane. The heuristic has been shown to generate graph layouts that follow generally accepted aesthetic criteria, such as minimizing edge crossings, distributing vertices evenly, and reflecting underlying symmetries. Straightforward implementation of the heuristic has time complexity $\Theta(V^2 + E)$ for each iteration, but there is an improvement that allows it to run in time $\Theta(V + E)$ per iteration.

The heuristic begins with an initial set of plane coordinates for the vertices (documents), usually random. The algorithm then iteratively applies three steps: (i) computing the effect on vertices by repulsive forces; (ii) computing the effect on vertices by attractive forces; and (iii) updating the temperature value. In computing forces, consider the ideal distance κ between vertices, which is

$$\kappa = C \sqrt{\frac{A_{\text{frame}}}{n}}.$$

Here A_{frame} is the area of the frame that is to contain the visualized graph, and C is an experimentally determined parameter for how the vertices fill the frame. In our experiments, we found that an explicit frame can be abandoned altogether, making C unnecessary and the value of κ arbitrary.

The magnitude of the repulsive force $f_r(d)$ between two vertices separated by distance d is then modeled as

$$f_r(d) = \frac{-\kappa^2}{d}.$$

At each iteration of the algorithm, this force is applied between all pairs of vertices. The repulsive force decreases at larger distances, and its repulsive nature is reflected by the negative sign. The magnitude of the attractive force $f_a(d)$ between the vertices is modeled as

$$f_a(d) = \frac{d^2}{\kappa}.$$

This force is applied only between adjacent vertices, i.e., those connected by an edge of the original graph. It provides an attractive force that increases quadratically with distance. The directions of the forces between two vertices are in a direct line to the other vertex.

Figure 2 shows the attractive and repulsive forces, along with their sum. This combined force is zero at the ideal distance $d = \kappa$. Thus edge distances in the layout tend to converge toward κ under iterations of the algorithm. An example convergence of the spring algorithm is shown in Figure 3.

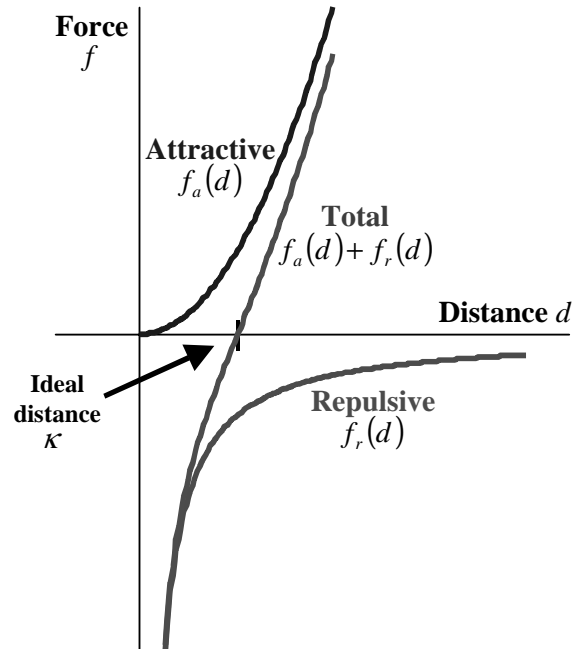


Figure 2. Attractive, repulsive, and total forces versus distance. Force cancel at ideal distance $d = \kappa$.

3. Visualization of co-citation MSTs

Pearson's correlation can be used to compute co-citation similarities. Correlations and co-citation counts are alike in that they are both dot products of a pair of

vectors (or generalized dot product for higher-order co-citations). The difference is that for correlations, the vectors are first converted to z scores (zero mean and unit standard deviation). Co-citation count has a more direct interpretation, as the actual number of documents that co-cite a pair.

Another difference between co-citation counts and correlations is that for a given document set, counts tend to have more repeated distance values than do correlations. This is because counts are independent of mean and variance. One could interpret correlations as having finer distance granularity.

A consequence of this is that the minimum spanning trees for distances computed from correlations tend to be more “chained.” That is, the average degree of a minimum spanning tree vertex tends to be lower for correlation-based distances. This causes edges were formerly incident on “central” vertices of count-based minimum spanning trees to be “stretched out” in chains over multiple vertices. Figures 4 to 6 show the visualization results using the SCI “Wavelets” data sets. These data sets are generated by the method described in Section 2, using the keyword “Wavelets.”

Our results show that count-based minimum spanning trees are less likely to become stuck in local optima during force-directed placement. This problem becomes more acute as the number of vertices increases. There are possible modifications to the placement algorithm to help avoid these local minima, such as by using simulated annealing, but the count-based minimum spanning trees are fundamentally easier to place. Count-based minimum spanning trees also make more efficient use of the placement area, in some sense “filling” more of the plane.

4. Concluding remarks

In this paper, we described an algorithm for computing spatial coordinates for the minimum spanning tree for the purpose of visualization. We compared the visualization of MST based on co-citation counts and on correlation. Our ongoing work includes applying the

hybrid pairwise/higher-order distances proposed in [7] to minimum spanning tree visualizations. The new distance methodology reduces distances among members of association mining frequent itemsets (higher-order co-citations). This should result in more direct influences among the frequent itemset members in the minimum spanning tree visualization, and should generally increase their influence within the entire tree.

Acknowledgments

This work was supported in part by the Louisiana Board of Regents’ Information Technology Research program.

References

- [1] R. Baiza-Yates, B. Ribeiro-Neto, *Modern Information Retrieval*, ACM Press, New York, 1999.
- [2] C. Chen, “Visualising Semantic Spaces and Author Co-Citation Networks in Digital Libraries,” *Information Processing & Management*, 35, 1999, pp. 401-420.
- [3] C. Chen, L. Carr, “Trailblazing the Literature of Hypertext: An author co-citation analysis (1989-1998),” in *Proceedings of the 10th ACM Conference on Hypertext (Hypertext '99)*, Darmstadt, Germany, February, 1999, pp. 51-60.
- [4] W. Venables, B. Ripley, *Modern Applied Statistics with S-Plus*, Springer-Verlag, Berlin, 1994.
- [5] P. Eades, “A Heuristic for Graph Drawing,” *Congressus Numerantium*, 42, 1984, pp. 149-160.
- [6] T. Fruchterman, E. Reingold, “Graph Drawing by Force-Directed Placement,” *Software – Practice and Experience*, 21, 1991, pp. 1129-1164.
- [7] S. Noel, V. V. Raghavan, C. H. Chu, “Visualizing association mining results through hierarchical clusters,” in *Proc. 2001 Int. Conf. Data Mining*, San Jose, Calif., Nov.-Dec. 2001, pp. 425-432.

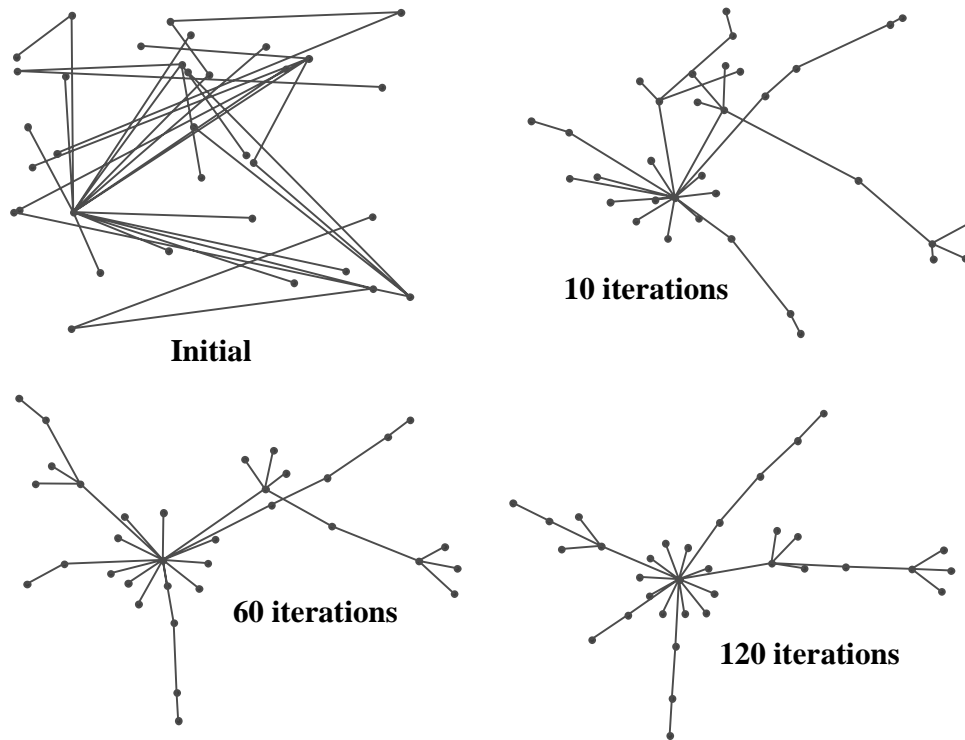


Figure 3. Iterations of spring algorithm for placing vertices of minimum spanning tree.

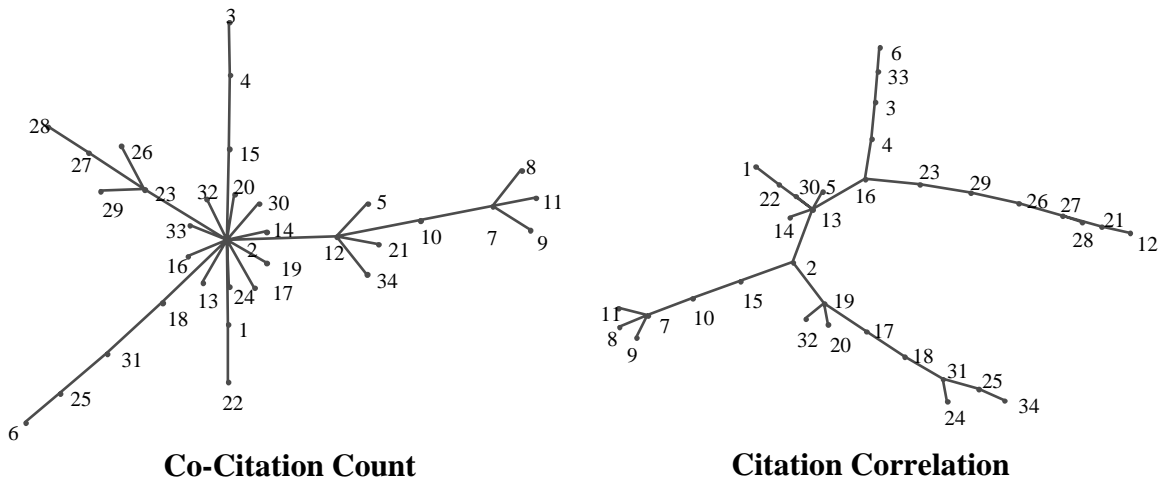
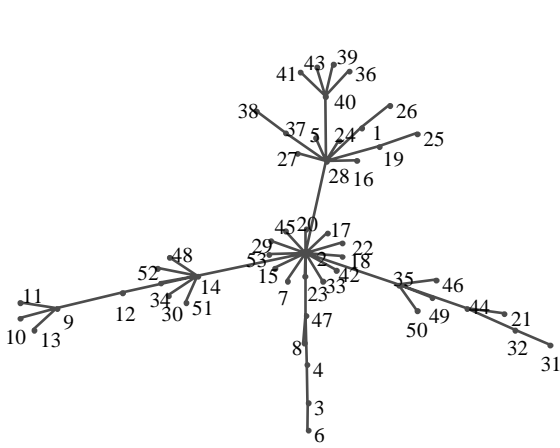
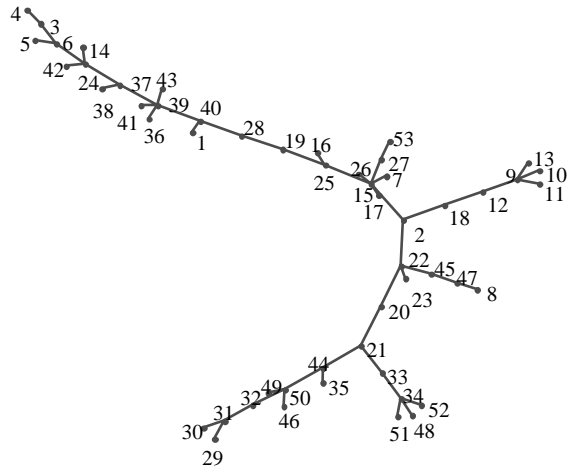


Figure 4. Minimum spanning tree placement for pairwise distances computed via co-citation count versus citation correlation, for data set “Wavelets 1999 (1-100).”

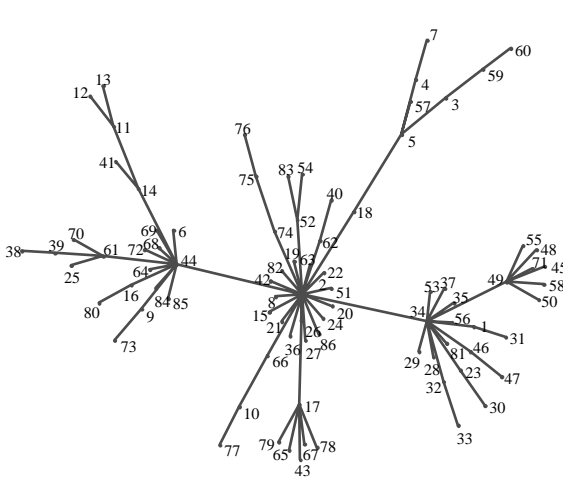


Co-Citation Count

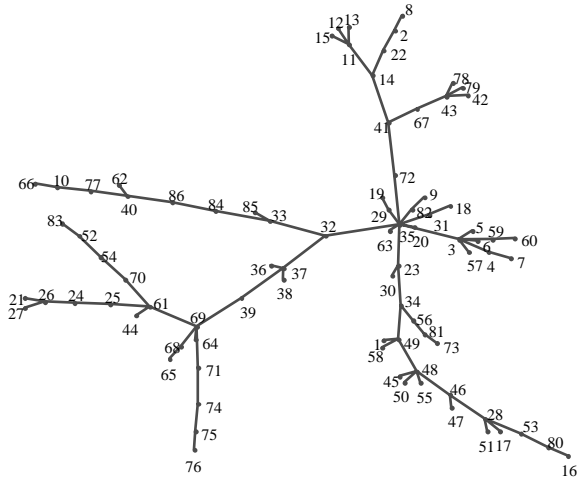


Citation Correlation

Figure 5. Minimum spanning tree placement for pairwise distances computed via co-citation count versus citation correlation, for data set “Wavelets 1999 (1-150).”



Co-Citation Count



Co-Citation Correlation

Figure 6. Minimum spanning tree placement for pairwise distances computed via co-citation count versus citation correlation, for data set “Wavelets 1999 (1-200).”