# Visual Ranking of Link Structures*
## (Extended Abstract)

Ulrik Brandes and Sabine Cornelsen

Department of Computer & Information Science, University of Konstanz, Box D 188, 78457 Konstanz, Germany. {Ulrik.Brandes,Sabine.Cornelsen}@uni-konstanz.de

**Abstract.** Methods for ranking World Wide Web resources according to their position in the link structure of the Web are receiving considerable attention, because they provide the first effective means for search engines to cope with the explosive growth and diversification of the Web. We show that layouts for effective visualization of an underlying link structure can be computed in sync with the iterative computation utilized in all popular such rankings. Our visualizations provide valuable insight into the link structure and the ranking mechanism alike. Therefore, they are useful for the analysis of query results, maintenance of search engines, and evaluation of Web graph models.

## 1 Introduction

The directed graph induced by the hyperlink structure of the Web has been recognized as a rich source of information. Understanding and exploiting this structure has a proven potential to help dealing with the explosive growth and diversification of the Web. Probably the most widely recognized example of this kind is the PageRank index employed by the Google search engine [6].

PageRank is but one of many models and algorithms to rank Web resources according to their position in a link structure (see, e.g., [25, 20, 9, 1, 5, 8]). Our goal is to supplement rankings with a meaningful visualization of the graph they are computed on.

While graph visualization is an active area of research as well [10, 19], its integration with quantitative analysis is only beginning to receive attention. It is, however, rather difficult to understand the determinants of a particular ranking if a visualization is prepared without explicitly taking it into account.

A design for graph visualizations showing vertex prominence in the structural context is introduced in [4], where the vertical dimension of the layout space is reserved to represent exactly the prominence of each vertex, but since horizontal layout is done by an adaptation of the Sugiyama framework for layered graph drawing [27] it does not scale to graphs with more than a few hundred vertices.

In the present application, it is also highly desirable that dense subgraphs be clustered, since on the Web they typically correspond to related resources.

* To appear in *Proc. 7th Intl. Workshop Algorithms and Data Structures (WADS '01)*, Lecture Notes in Computer Science. Springer, 2001.

A well-known class of graph layout algorithms that exhibit these properties are spring embedders, and the range for which they are practical has recently been extended to sparse graphs with several thousands of vertices [13, 15, 29]. Since they require relatively complex memory management, however, they are less suitable in our situation.

Instead, we apply a spectral graph layout approach, because of its formal and computational similarities with common ranking techniques. Background on link-based ranking is given in Sect. 2. The actual layout definition and computation are described in Sect. 3. In Sect. 4, we discuss applications and provide examples on generated and real-world data.

## 2 Structural Ranking of Web Resources

The structural features of the Web are captured in a directed graph $G = (V, E)$, where the set $V$ of vertices represents the set of resources on the Web, and there is a directed edge $(u, v) \in E$ from a resource $u$ to a resource $v$, if $u$ contains a hyperlink to $v$. All graphs considered in this paper are assumed to be connected. We do not allow parallel edges, but loops and a positive real weight $\omega_{uv}$ for every edge. Let $A(G) = A = (A_{uv})_{u,v \in V}$ be the *adjacency matrix* of a graph, i.e. $A_{uv} = \omega_{uv}$ if $(u, v) \in E$, and $A_{uv} = 0$ otherwise. The *indegree* (*outdegree*), $d_v^+$ ($d_v^-$), of a vertex $v \in V$ is $\sum_{u:(u,v)\in E} A_{uv}$ ($\sum_{w:(v,w)\in E} A_{vw}$).

We will frequently use graph-related matrices, such as the adjacency matrix, and their *spectra*, i.e. the set of eigenvalues and associated eigenvectors. For background on matrix computations see, e.g., [14].

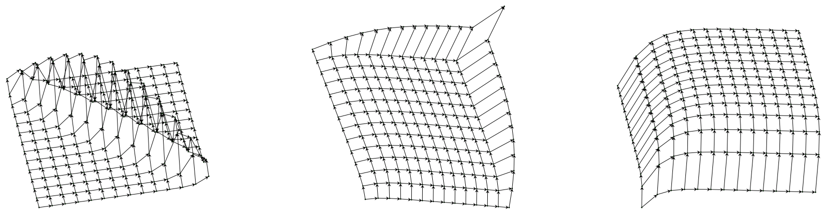
Any real-valued vector $p = (p_v)_{v \in V}$ defined on the vertices of a graph is called a *prominence index*, where $p_v$ is the *prominence* of vertex $v$. A *ranking* is obtained from a prominence index by ordering the vertices according to non-increasing prominence.

Many models have been proposed to capture an explicitly or implicitly defined notion of a vertex's prominence in a graph [18, 17, 11, 3, 12, 25, 20, 1, 9, 8, and many more]. Though in general only defined for undirected graphs, we first outline *eigenvector centrality* [2], because it nicely illustrates some important commonalities of the three popular rankings that we discuss below.

Assume that the prominence of a vertex is understood to be proportional to the combined prominence of its neighbors, $\lambda p_v = \sum_{u:\{u,v\}\in E} \omega_{uv} p_u$, $v \in V$, where the constant $\lambda$ is introduced so that the system of equations has a non-zero solution. This definition yields the eigensystem of the transposed adjacency matrix,

$$\lambda p = Ap, \qquad \text{(\textit{eigenvector centrality})}$$

and every eigenvector of $A$ gives a ranking of the vertices for the above notion of prominence, though the *principal eigenvector*, i.e. the one associated with the eigenvalue of largest magnitude, is generally preferred [3, 12]. The principal eigenvector can be obtained by power iteration, which starts with any non-zero vector and iteratively multiplies the matrix with the current solution, e.g.



(a) authority (b) PageRank (c) Katz's status

**Fig. 1.** Prominence indices on a directed grid

$p^{(0)} \leftarrow \mathbf{1}$ and

$$p^{(k+1)} \leftarrow A \cdot p^{(k)}.$$

Since the matrices considered here originate from large and sparse graphs, multiplication is carried out by computing $p_v^{(k+1)} \leftarrow \sum_{u:\{u,v\}\in E} p_u^{(k)}$ for every $v \in V$. In this and in the iterations to follow, we tacitly assume that each vector $p^{(k)}$ is normalized (e.g., to the length of the starting vector) before the next vector is computed. This serves to avoid numerical difficulties with numbers growing out of range, and does not affect the relative prominence of vertices.

More elaborate indices defined on directed graphs are discussed below. In Fig. 1 they are illustrated on an acyclic grid. The grid is placed in a plane and each grid point is then lifted according to its prominence.

*Hubs and authorities [20].* A natural notion of prominence for a Web resource is the extent to which it is referred to by other Web pages, in particular by those pages that specialize in listing useful resources. In turn, the property of being such a list of useful resources is a notion of prominence in itself. In these complementary and mutually reinforcing notions prominent resources are called *authorities* (resources with useful information) and *hubs* (pages with useful links).

The hub score of a page is proportional to the combined authority of the resources it links to, and the authority of a resource is proportional to the combined hub score of the pages linking to it. In practice, hub and authority scores are thus computed by iterating $p^{(0)} \leftarrow \mathbf{1}$ and

$$\begin{aligned} p^{(2k+1)} &\leftarrow A^T \cdot p^{(2k)} \\ p^{(2k+2)} &\leftarrow A \cdot p^{(2k+1)}. \end{aligned}$$

For $h^{(k)} = p^{(2k)}$ and $a^{(k)} = p^{(2k+1)}$, the alternating iteration can be written as

$$h^{(k+1)} \leftarrow AA^T \cdot h^{(k)} \qquad (\textit{hubs})$$

$$a^{(k+1)} \leftarrow A^TA \cdot a^{(k)}. \qquad (\textit{authorities})$$

In this formulation, it is easy to see that the hub and authority indices in a graph with adjacency matrix $A$ correspond to eigenvector centrality in the weighted undirected graphs with adjacency matrix $AA^T$ and $A^TA$, respectively.

As can be seen in Fig. 1(a), vertices on and above the falling diagonal of the grid have the highest authority, because they are in the midst of the undirected graph induced by $A^TA$. Compare this to the undirected graph induced by $AA^T$, indicating why the best hubs are found on and below this diagonal.

*PageRank [5].* In another variant of eigenvector centrality the contribution of each vertex to another vertex's prominence is weighted by its outdegree, $p_v = \sum_{u:(u,v)\in E} \frac{p_u}{d_u^-}$ (see e.g. [24, 7]). If we require $p$ to be a probability distribution over the set of vertices, this notion has a nice interpretation as the stationary distribution of the simple random walk on the graph (or random surfer on the Web, if you will), in which each edge leaving a vertex is chosen with equal probability.

Let $M = D_-^{-1}A$ be the adjacency matrix normalized so that the rows sum to one, where $D_-$ is the diagonal matrix with the outdegrees on the diagonal. Then, $M$ is a stochastic matrix of transition probabilities, and a stationary distribution $p = M^T \cdot p$ satisfies the above notion of prominence. However, if a vertex has outdegree zero, the computation breaks down, and strongly connected components may cause an overdue increase of the prominence of their vertices. This so-called "sink problem" can be avoided by introducing an escape mechanism. Let $\hat{p}$ be an a-priori probability distribution over the vertices (e.g., user preferences or general popularity of a resource), then with probability $\omega$ the random walk picks an edge of the graph whereas with the remaining probability, it jumps to any other vertex according to $\hat{p}$. The index is thus defined by

$$\begin{aligned} p &= \omega M^T p + (1-\omega)\hat{p} \qquad (\textit{PageRank}) \\ &= \left(\omega M^T + (1-\omega)\hat{p}\cdot \mathbf{1}^T\right)\cdot p. \end{aligned}$$

The second equality holds because $p$ is a probability distribution. From the second expression it can be seen that PageRank is the eigenvector centrality of a weighted graph with a complete set of additional escape edges. This modified matrix is irreducible and aperiodic so that the iteration $p^{(0)} \leftarrow \frac{1}{n}\mathbf{1}$ and

$$p^{(k+1)} \leftarrow \left(\omega M + (1-\omega)\mathbf{1}\cdot\hat{p}^T\right)^T \cdot p^{(k)}$$

converges to a unique prominence vector. On the grid in Fig. 1(b), the random surfer may jump to any vertex, but is most likely to walk towards the upper and right side of the grid, from where the only continuation is towards the upper right corner.

*Katz's status index [18].* As a generalization of simply using indegrees to measure 'status' in social networks, the prominence of a vertex is determined by the number of directed paths of arbitrary length ending in the vertex, where the influence of longer paths is attenuated by a decay factor. Recall that the entries of the $k$-th power of the adjacency matrix of an unweighted graph give the number of paths of length $k$ between every pair of vertices. Therefore, this notion of prominence is determined by

$$p = \left( \sum_{k=1}^{\infty} (\alpha A^T)^k \right) \cdot \mathbf{1}, \qquad \textit{(Katz's status)}$$

where parameter $\alpha$ corresponds to the fraction of status that is passed along a directed edge. For sufficiently small values of $\alpha$ (a convenient choice is $\frac{1}{\Delta^+ + 1}$, where $\Delta^+$ is the maximum indegree of any vertex in the graph), the sum converges to $(I - \alpha A^T)^{-1} - I$. Therefore, the status vector can be obtained by solving $\left(\alpha^{-1} I - A^T\right) \cdot p = d^+$. Solving this system of linear equations directly is prohibitive for large graphs. Standard sparse matrix approaches approximate a solution iteratively. The update step in Jacobi iteration, for instance, yields $p^{(k+1)} \leftarrow \alpha A^T \cdot p^{(k)} + \alpha d^+$. This iteration nicely reflects the underlying notion of adding contributions from vertices farther and farther away. The same can be observed in Fig. 1(c), where the attenuated influence from vertices in the lower left does not suffice to discriminate the prominence of vertices in the upper right any more.

In a sense, the above definitions of prominence are contained in the following generic formulation of status in networks [17]. It puts a twist on eigenvector centrality through the addition of an a-priori prominence vector $\hat{p}$,

$$p = A^T p + \hat{p}. \qquad \textit{(Hubbel's status)}$$

By choosing appropriate weights and a-priori prominences, we obtain eigenvector centrality and PageRank. Reordering, we have $p = (I - A^T)^{-1} \cdot \hat{p}$, provided the inverse exists. If it does, it equals $\sum_{k=0}^{\infty} (A^T)^k$, and therefore $p = \left(\sum_{k=0}^{\infty} (A^T)^k\right) \cdot \hat{p} = \left(I + \sum_{k=1}^{\infty} (A^T)^k\right) \cdot \hat{p}$. With uniform edge weights and $\hat{p} = \mathbf{1}$ we obtain a prominence index in which every component is by one larger than Katz's status index.

## 3 Spectral Graph Layout

In the previous section we emphasized formal similarities in the definition of several popular prominence indices. In practice, all of them are computed by some variant of sparse matrix power iteration, i.e. by iterating over all vertices, and, for each vertex, combining the current scores of its neighbors. For really large graphs, most of the running time is spent on data transfer between internal and external memory. It is thus desirable not to cause additional swapping.

In this section, we introduce a layout algorithm that produces meaningful layouts while synchronously operating on the same data that prominence implementations do, because it is based on the same principles as the ranking algorithms. It is therefore a simple matter to augment an existing system for ranking resources to compute a layout of the graph on the fly.

*Layout with eigenvectors.* For the layout, we consider the undirected, simple graph obtained by omitting directions, loops, and multiple edges. Recall that directions are already represented in the prominence dimension. Let $A$ be the adjacency matrix of the skeleton and $D = D^+ = D^-$ its diagonal *degree matrix*. We consider the *Laplacian matrix* $L = D - A$, which has interesting applications in many areas (see, e.g., [23]). This matrix is interesting for graph layout, since minimizing the associated quadratic form

$$x^T L x = \sum_{\{u,v\} \in E} (x_u - x_v)^2,$$

corresponds to minimizing the squared distance between pairs of adjacent vertices, if $x$ is interpreted as a vector of vertex positions. This corresponds to spring embedding with zero-length springs and no repelling forces. Clearly, the minima are obtained by assigning the same position to all vertices (recall that we assume connectedness of the graph).

These undesirable solutions can be avoided by fixing some selected vertices at distinct positions. Minimization subject to these boundary conditions is the famous *barycentric layout* model of Tutte [28]. However, it requires some a-priori knowledge about which vertices should be fixed where.

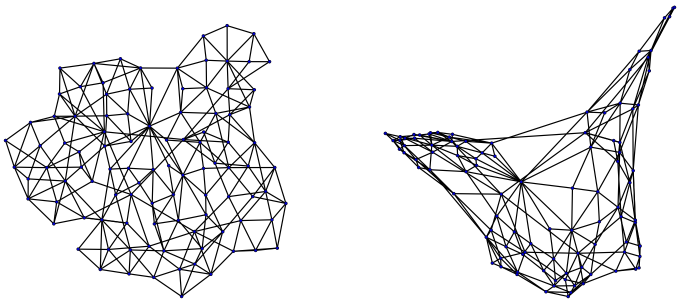
Note that the undesired minima $x = c\mathbf{1}$ are the eigenvectors associated with eigenvalue zero, i.e. $Lx = 0$. More generally, if $(\lambda, x)$ is any eigenpair of $L$, then $\lambda = \frac{x^T L x}{x^T x}$. We therefore want to minimize

$$\frac{x^T L x}{x^T x} \text{ subject to } x \perp \mathbf{1},$$

since the eigenvectors of a symmetric matrix are orthogonal. Hence, the desired solution is the normalized eigenvector associated with the second-smallest eigenvalue of $L$. This vector is called the *Fiedler vector* and, because of its distance minimization property, often used in graph partitioning (see, e.g., [26]). For the same reason, it yields a useful one-dimensional layout of a graph, because edges are short and hence dense subgraphs are clustered. If a rank visualization in three dimensions is desired (cf. Fig. 1), a reasonable choice for the second free dimension is the normalized eigenvector minimizing the objective function subject to being orthogonal to $\mathbf{1}$ and the first solution.

An example of two-dimensional layouts obtained by a typical spring embedder and two eigenvectors of $L$ is given in Fig. 2. While the spring embedder produces more uniform edge lengths, the eigenvectors emphasize structural clustering of vertices.

*Computing the eigenvectors.* Eigenvectors associated with the smallest eigenvalues of large sparse matrices are usually computed using Lanczos' method. However, all popular prominence indices are computed using a variant of the much simpler power iteration, which only gives an eigenvector associated with the eigenvalue of largest magnitude. Since we want to synchronize layout and prominence computation, we consider the matrix $L' = 2\Delta \cdot I - L$ instead of $L$



(a) spring embedder (b) spectral layout

**Fig. 2.** Two-dimensional layouts of a random planar triconnected graph

itself, where $\Delta$ is the maximum degree of any vertex. The crucial observation is that the eigenvectors of $L$ and $L'$ are identical, but the order of the magnitudes of their corresponding eigenvalues is reversed.

Straightforward application of power iteration on $L'$ returns the principal eigenvector of $L'$, which is the trivial eigenvector $\frac{1}{\sqrt{n}}\mathbf{1}$ of $L$. Power iteration on a vector that is orthogonal to the principal eigenvector yields an eigenvector of the second-largest eigenvalue of $L'$, and hence the desired layout for the first dimension. Iterating on a vector that is orthogonal to both the trivial eigenvector and the approximate solution for the first dimension yields the second dimension.

A vector $y$ is orthogonalized with respect to another vector $x$ by setting $y \leftarrow y - \frac{x^T \cdot y}{x^T \cdot x}\, x$. Orthogonalization with respect to the trivial eigenvector $\frac{1}{\sqrt{n}}\mathbf{1}$ is even easier, since it corresponds to subtracting, from each entry of $y$, the mean of all its entries. To obtain vectors $x$ and $y$ for a two-dimensional layout we thus carry out the following augmented power iteration on random starting vectors $x^{(0)}, y^{(0)}$ that are repeatedly orthogonalized with respect to $\mathbf{1}$ and to one another

$$
\begin{aligned}
x^{(k+1)} &\leftarrow L' \cdot x^{(k)}; \qquad & x^{(k+1)} &\leftarrow x^{(k+1)} - \frac{1}{n}\sum_{v \in V} x_v^{(k+1)} \\
y^{(k+1)} &\leftarrow L' \cdot y^{(k)}; \qquad & y^{(k+1)} &\leftarrow y^{(k+1)} - \frac{1}{n}\sum_{v \in V} y_v^{(k+1)} \\
y^{(k+1)} &\leftarrow y^{(k+1)} - \frac{x^{(k+1)T} \cdot y^{(k+1)}}{x^{(k+1)T} \cdot x^{(k+1)}}\, x^{(k+1)}
\end{aligned}
$$

Intuitively, the layout is centered, rectified, and (due to the tacitly assumed normalization) zoomed after each multiplication with $L'$. The last two lines are omitted if only one dimension needs to be determined for the layout. Though convergence may be slower than in the prominence computations, a few iterations are usually sufficient for a reasonable layout.

## 4 Application to Web Graph Models and Query Results

We demonstrate our visualization approach on two different kinds of data, random Web graphs generated according to slightly modified versions of the *evolving copying models* of [21] and the *small-world model* of [30], and a real-world example obtained from an AltaVista query. Our C++-implementations use the *Library of Efficient Data Types and Algorithms* (LEDA) [22].

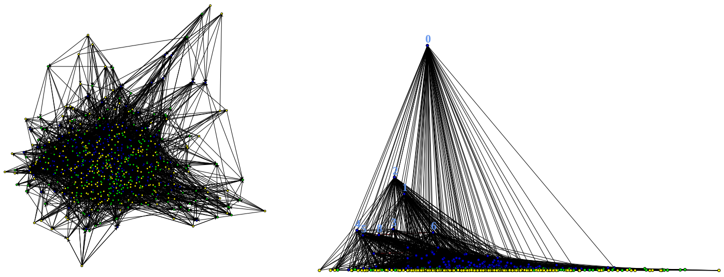
*Web graph models.* In the *linear growth model*, a graph grows one vertex at a time. At each time step, a prototype is chosen among already existing vertices, and a new vertex is generated. This new vertex is then assigned a fixed number of outgoing edges. With some fixed probability, the $i$th of these edges points to a randomly selected vertex among those already existing (creation case), and with the remaining probability it points to the same vertex as the $i$th outgoing edge of the prototype vertex (copying case). Our generator does not introduce multiple edges, and if a prototype happens to not have enough outgoing edges, no edge is introduced in the copying case. Clearly, all graphs evolving like this are acyclic.

In the *exponential growth model*, a graph grows by a fixed fraction of its current size at each time step. New vertices receive a fixed number of loops, and for each already existing edge, its target receives a new incoming edge for which, with some fixed probability, the source is chosen uniformly at random from the new vertices, and otherwise from the existing vertices with probability proportional to their current outdegree. We used a simpler model in which existing vertices are chosen uniformly at random as well.

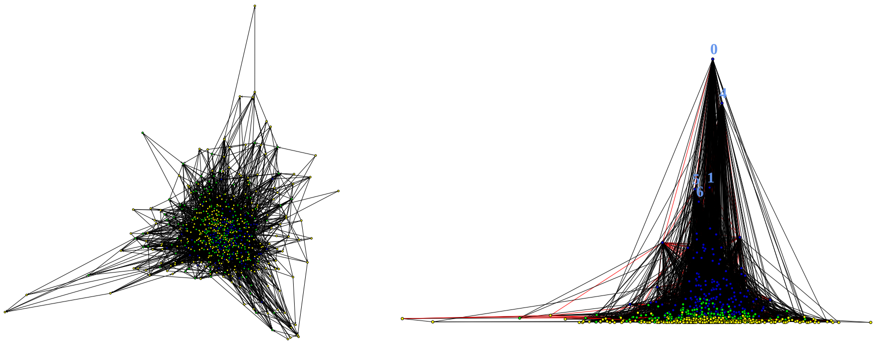
In the *small-world model*, we initially generate a cyclic sequence of vertices and let a vertex link to a fixed number of predecessors and successors. Then, each edge is rewired with some small probability by chosing a new destination uniformly at random.

Figure 3 shows spectral layouts of graphs generated according to these models and rankings replacing the vertical dimension with PageRank as an example of a prominence index. There is no visible clustering in the evolving copying models. Moreover, the prominence of resources appears to be correlated with their age (also with the other indices outlined in Sect. 2). The figures thus graphically support the conclusion of [21] that *death processes*, i.e. the occasional deletion of vertices and edges, might be necessary for the evolving copying models to be realistic. In the small-world model, the spectral layout reveals a cycle crumpled by chords, and the ranking shows that the model yields a rather egalitarian structure.

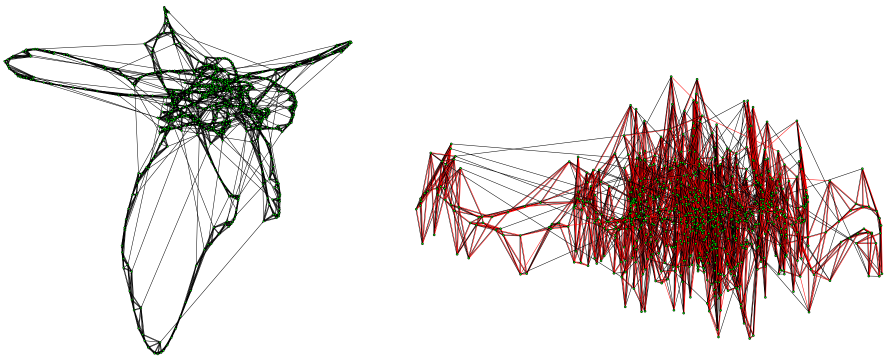
*Query results.* The data for this example was compiled in a way similar to the HITS algorithm [20]. We asked the AltaVista search engine for pages containing the word "java" and used the first 200 URLs it returned as the root set. It was then expanded by asking AltaVista for pages containing links to resources in the root set (backward extension), and adding resources linked to by pages in the root set (forward extension). The graph was completed by adding edges



(a) Linear growth evolving copying model [21]



(b) Exponential growth evolving copying model [21]



(c) Small-world model [30]

**Fig. 3.** Web graph models (2D spectral layout and 1D spectral layout with PageRank)

between pages in the resulting set of vertices. The computations were carried out on the only large component of this graph from which some poorly connected vertices were removed to prevent extreme clustering. The graph has more than 5000 vertices and 15000 edges.

In Fig. 4, this graph is shown twice, with vertices positioned vertically according to the Fiedler vector, and horizontally according to one of two prominence indices. Again, links to less prominent resources are colored red.

The most prominent resources match the expectations, but there are some less expected recommendations as well. It is clearly visible that some of these serve distinct user groups, like the japanese directory in the upper right. Note that we may not conclude that vertically close vertices are closely connected without zooming into the image. However, it is safe to assume that vertically separated vertices are likely to be distant in the structure. This feature can serve to distinguish query results which contain a keyword that is used in different contexts (see the "jaguar"-query example in [20]).

Figure 4 also shows that the top authorities are surprisingly distinguished from the rest of the graph, and quite different from our expectations. Most of them are located at `Stars.com`, a large repository for developers ("Web Developer's Virtual Library"). Since they are well connected among each other, it is by virtue of our layout approach that their vertical position is similar, and thus this phenomenon could be detected by visual exploration. In the figure, resources at this site are colored yellow. Not surprisingly, vertices with high hub scores are from this site as well. This simple example graphically explains why the original HITS algorithm does not consider links within a site.

## 5 Conclusions

We have proposed a method for Web graph visualization that provides unambiguous identification of prominent resources while showing the entire graph and its clustering. The layout of our visualizations can be computed synchronously with common link-based rankings. Speed-up techniques such as in [16] that reorganize storage to reduce external memory access therefore directly apply to the layout algorithm as well.

We expect our visualizations to be particularly useful for visual inspection of rankings, for teaching and communicating ranking procedures, and for evaluation and illustration of stochastic models of the Web graph. The main advantage of spectral graph layout, the correspondence with distance minimization and hence with clustering, becomes a drawback in cases where the underlying undirected graph is poorly connected, since denser subgraphs will be clustered in a very small interval. This problem is addressed in the full version of this paper, which also contains an analysis of the convergence properties of the iterative computation.

*Acknowledgments.* We thank Marco Gaertler for collecting the "java"-query data used in Sect. 4, and Stephen Muth for discussions regarding the iterative computation of Katz's status index.

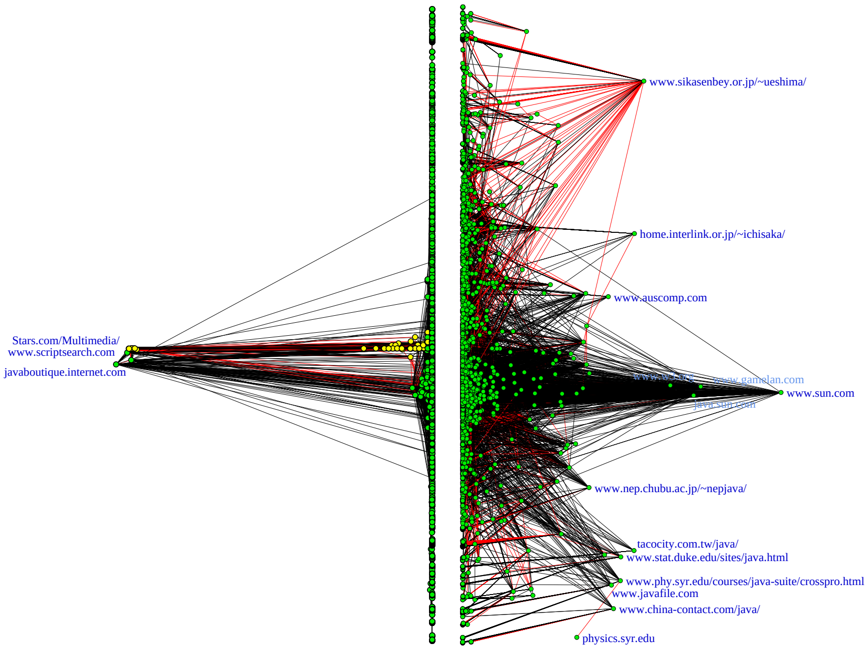



**Fig. 4.** Authority and PageRank visualization of "java" query result

## References


1. K. Bharat and M. R. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *Proc. 21st Ann. Intl. ACM SIGIR Conf. Research and Development in Information Retrieval*, pp. 104–111, 1998.
2. P. Bonacich. Factoring and weighting approaches to status scores and clique identification. *Journal of Mathematical Sociology*, 2:113–120, 1972.
3. P. Bonacich. Power and centrality: A family of measures. *American Journal of Sociology*, 92:1170–1182, 1987.
4. U. Brandes and D. Wagner. Contextual visualization of actor status in social networks. In *Data Visualization 2000. Proc. 2nd Joint Eurographics and IEEE TCVG Symp. Visualization*, pp. 13–22. Springer, 2000.
5. S. Brin, R. Motwani, L. Page, and T. Winograd. What can you do with a web in your pocket? *IEEE Bulletin of the Technical Committee on Data Engineering*, 21(2):37–47, June 1998.
6. S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.
7. R. S. Burt. *Toward a Structural Theory of Action: Network Models of Social Structure, Perception, and Action.* Academic Press, 1982.
8. S. Chakrabarti, B. E. Dom, R. Kumar, P. Raghavan, S. Rajagopalan, A. S. Tomkins, D. Gibson, and J. M. Kleinberg. Mining the Web's link structure. *IEEE Computer*, 32(8):60–67, 1999.

9. S. Chakrabarti, B. E. Dom, P. Raghavan, S. Rajagopalan, D. Gibson, and J. M. Kleinberg. Automatic resource compilation by analyzing hyperlink structure and associated text. *Computer Networks and ISDN Systems*, 30(1–7):65–74, 1998.
10. G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs.* Prentice Hall, 1999.
11. L. C. Freeman. Centrality in social networks: Conceptual clarification. *Social Networks*, 1:215–239, 1979.
12. N. E. Friedkin. Theoretical foundations for centrality measures. *American Journal of Sociology*, 96(6):1478–1504, 1991.
13. P. Gajer, M. T. Goodrich, and S. G. Kobourov. A fast multi-dimensional algorithm for drawing large graphs. In *Proc. 8th Intl. Symp. Graph Drawing*, LNCS 1984:211–221. Springer, 2001.
14. G. H. Golub and C. F. van Loan. *Matrix Computations.* Johns Hopkins University Press, 3rd ed., 1996.
15. D. Harel and Y. Koren. A fast multi-scale method for drawing large graphs. In *Proc. 8th Intl. Symp. Graph Drawing*, LNCS 1984:183–196. Springer, 2001.
16. T. H. Haveliwala. Efficient computation of pagerank. Technical Report 1999-31, Database Group, Stanford University, 1999.
17. C. H. Hubbell. An input-output approach to clique identification. *Sociometry*, 28:377–399, 1965.
18. L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18:39–43, 1953.
19. M. Kaufmann and D. Wagner (eds). *Drawing Graphs: Methods and Models*, LNCS Tutorial 2025. Springer, 2001.
20. J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.
21. R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. S. Tomkins, and E. Upfal. Stochastic models for the Web graph. In *Proc. 41st Ann. IEEE Symp. Foundations of Computer Science*, pp. 57–65, 2000.
22. K. Mehlhorn and S. Näher. *The LEDA Platform of Combinatorial and Geometric Computing.* Cambridge University Press, 1999.
23. B. Mohar. Some applications of Laplace eigenvalues of graphs. In *Graph Symmetry.* NATO ASI Series C 497, pp. 225–275. Kluwer, 1997.
24. G. Pinski and F. Narin. Citation influence for journal aggregates of scientific publications: Theory, with applications to the literature of physics. *Information Processing and Management*, 12(5):297–312, 1976.
25. P. Pirolli, J. Pitkow, and R. Rao. Silk from a sow's ear: Extracting usable structures from the Web. In *Proc. ACM Conf. Human Factors in Computing Systems*, pp. 118–125, 1996.
26. D. A. Spielman and S.-H. Teng. Spectral graph partitioning works: Planar graphs and finite element meshes. In *Proc. 37th Ann. IEEE Symp. Foundations of Computer Science*, pp. 96–105, 1996.
27. K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man and Cybernetics*, 11(2):109–125, 1981.
28. W. T. Tutte. How to draw a graph. *Proceedings of the London Mathematical Society, Third Series*, 13:743–768, 1963.
29. C. Walshaw. A multilevel algorithm for force-directed graph drawing. In *Proc. 8th Intl. Symp. Graph Drawing*, LNCS 1984:171–182. Springer, 2001.
30. D. J. Watts and S. H. Strogatz. Collective dynamics of "small-world" networks. *Nature*, 393:440–442, 1998.