Contextual Visualization of Actor Status in Social Networks

Ulrik Brandes* and Dorothea Wagner

University of Konstanz, Department of Computer and Information Science, Box D 188, 78457 Konstanz, Germany
{Ulrik.Brandes,Dorothea.Wagner}@uni-konstanz.de

Abstract. We propose a novel information visualization approach for an analytical method applied in the social sciences. In social network analysis, social structures are formally represented as graphs, and structural properties of these graphs are assumed to be useful in the explanation of social phenomena. A particularly important such property is the relative status of actors in a given network. Since operationalizations of status are aggregate indices of vertices, researchers are not only interested in status, but also in the context leading to these values, i.e. the underlying social network. We therefore visualize the network in a layered fashion, mapping status scores to vertical coordinates. The resulting problem of determining horizontal positions of vertices such that the overall layout is readable, is algorithmically difficult, yet well-studied in the literature on graph drawing. We outline a customized approach that routinely produces satisfactory pictures at interactive speed.

1 Introduction

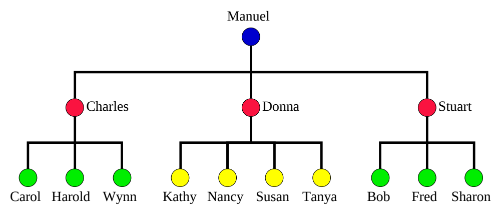
Different from categorical data analysis, aggregate indices of relational data are typically insufficient to fully appreciate and understand the information contained in the data. In any kind of network analysis, it is therefore desirable to always provide a representation of the actual network as well.

Most types of networks are traditionally visualized using point-and-line representations [4]. If the network has no underlying spatial layout (unlike, e.g., data associated with geographic networks [3]), a layout has to be computed explicitly. But in addition to the inherent difficulty of laying out an abstract network in a readable way [8], this raises the problem of trust in its analysis. Who is going to comfortably interpret complex aggregate data, when it is difficult to relate it to the base data? As a potential remedy for this problem we propose contextual visualization, i.e. the simultaneous representation of base and derived data in a single diagram that is based on some express principles. Simple examples of this principle are found, e.g., in bar charts with an additional line indicating the mean value. We here pursue this idea in an application from the social sciences.

* Part of this research was done while with the Department of Computer Science at Brown University. I gratefully acknowledge the *German Academic Exchange Service* (DAAD/Hochschulsonderprogramm III) for financial support.

Social network analysis is a subdiscipline of the social sciences, using graph-theoretic concepts to understand and explain social phenomena. A social network consists of a set of actors, who may be arbitrary entities like persons or organizations, and one or more types of relations between them. For a comprehensive overview of methods and applications see [29]. We here confine ourselves to networks of a single, directed relation.

The concept is illustrated by an example group of 14 employees, the internal auditing staff of a larger company. This group is analyzed in [17], where its formal organization is compared to an informal relation called “advice”, i.e. who does an actor turn to for help or advice at work about work-related questions or problems. Organizational and advice relation data are given in Fig. 1.



Manuel	0	0	0	0	0	0	0	0	0	0	0	1	0	0	*manager*
Charles	1	0	0	1	0	0	0	0	0	0	0	1	0	0	
Donna	1	0	0	0	0	0	0	0	0	0	0	1	0	0	*supervisors*
Stuart	1	1	0	0	0	0	0	0	0	0	0	1	0	0	
Bob	0	0	0	1	0	0	1	0	1	0	0	0	0	0	
Carol	0	1	0	0	0	0	0	0	0	0	0	0	0	0	
Fred	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
Harold	0	1	0	0	0	0	0	0	0	0	0	0	0	0	*auditors*
Sharon	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
Wynn	0	1	0	0	0	0	0	0	0	0	0	0	0	0	
Kathy	0	0	1	0	0	0	0	0	0	0	0	1	0	1	
Nancy	0	0	1	0	0	0	0	0	0	0	0	0	0	0	
Susan	0	0	1	0	0	0	0	0	0	0	1	0	0	1	*secretaries*
Tanya	0	0	1	0	0	0	0	0	0	0	1	1	0	0	

Fig. 1. Formal organizational chart and adjacency matrix of advice relationship. If a matrix entry is 1, the row actor turns to the column actor for advice

The advice relation largely resembles the organizational hierarchy with one notable exception, as illustrated in Fig. 2. In a tiresome and error-prone[1] process, vertices were manually arranged such that most edges point in upward direction, thus depicting an informal status in the advice network. Based on this graphical support, the conclusion of [17] is that changes the manager introduced to increase through-put may have been ineffective because he had not made sure that the secretary presiding the informal hierarchy of advice was backing them.

Qualitative results like this can be supported routinely using the formal apparatus of social network analysis. Networks of relationships are conveniently modeled by graphs $G = (V, E)$, where vertex set V represents the set of actors, and the set $E \subseteq V \times V$ of directed edges represents the relation under study, i.e., in our example, $(u, v) \in E$, if and only if the actor represented by u turns to the actor represented by v for advice. For convenience, we usually omit the distinction between actors and vertices, or edges and relations.

A simple, yet crude, quantitative measure of an actor’s network status is its indegree, defined as the number of edges directed to the vertex. Since this definition takes into account only status gained from direct links, several approaches have been developed to include also indirect links. To convey the flavor of these approaches, a commonly used definition of status is presented. Introduced in [16], it rests on the assumption that links from actors that have high status themselves contribute more to a receiving

[1] This is by necessity. See the paragraph on layer assignment in Section 3.

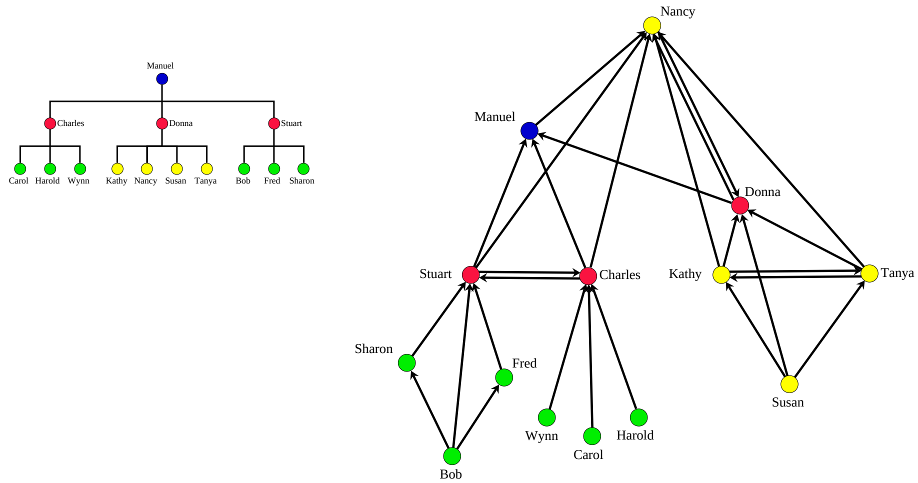


Fig. 2. Advice network, *manually* arranged so that most edges point upward (redrawn from [17])

actor's status than links from others. This recursive definition leads to the following equilibrium equation. Let $a < 1$ be an attenuation factor indicating the decrease of status passed along edges in the graph. If A denotes the adjacency matrix of the graph, then a solution $s = (s_v)_{v \in V}$ of

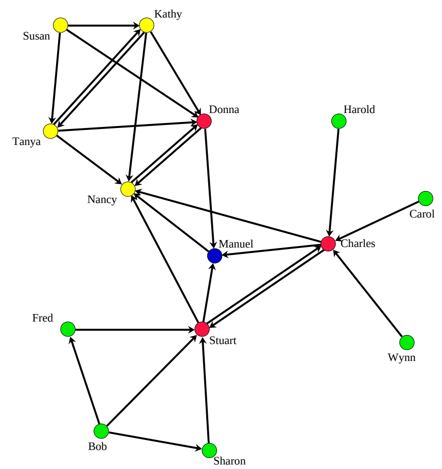
$$\left(\frac{1}{a} \cdot I - A^T\right) \cdot s = d^-,$$

where I is the unit matrix, A^T the transpose of A, and d^- the vector of indegrees, describes the relative status according to the above model. For ease of comparison, each entry in s is divided by the maximum entry. Status results for the example network are given in the next section. We refer the reader to Chapter 5 of [29] for references on sociological interpretation and other models of status.

The goal of this work is to provide automatic visual support for status analysis in social networks. Such work has three main aspects [5]: the substance to be visualized, a graphical design, and an algorithm realizing it. We have already described the substance we are interested in. The remainder of this paper is therefore organized as follows. In Sect. 2, we develop a graphical design for contextual visualization of networks and status therein, and give algorithms to produce such drawings in Sect. 3. We conclude in Sect. 4 with visualizations from a sociological study using a prototype implementation of our approach.

2 Contextual Visualization

In this section, we develop a graphical design for visualizations that contextualize status values with the underlying network. Currently available tools for network visualization[2] do not achieve this, because they essentially produce general purpose visualizations focusing on the ease of perceiving connectedness information, i.e. the presence or absence of edges between pairs of vertices. Figure 3 shows the type of visualization thus typically encountered. The network is shown separate from the result of the analysis (here, status according to the measure described above). Though the image is very readable, it does not convey the interesting status information. Its design is inherently undirected (the picture would be the same even if all edge directions are reversed), and it is next to impossible to relate the status values to the picture. Such visualizations are typical in the work of sociologists, and others applying their concepts (see, e.g., [13]).



1.00	Nancy	(secretary)
0.66	Donna	(supervisor)
0.57	Manuel	(manager)
0.19	Stuart	(supervisor)
0.17	Charles	(supervisor)
0.08	Kathy	(secretary)
	Tanya	(secretary)
0.02	Fred	(auditor)
	Sharon	(auditor)
0.00	Bob	(auditor)
	Carol	(auditor)
	Harold	(auditor)
	Wynn	(auditor)
	Susan	(secretary)

Fig. 3. Non-contextual automatic visualization of status and advice network (spring embedder type layout and stem-and-leaf diagram)

Empirical evidence suggests that network layout affects not only the ease of reading [25], but has an influence on the understanding and interpretation of substantive content as well [21]. Building on the familiar everyday notion of “higher” and “lower” status, it seems natural to graphically represent status through vertical positioning. Instead of using bar charts to depict status, the placement of vertices themselves can be restricted to levels signaling their status. The idea is illustrated in Fig. 4, where the status index described in the introduction is used to assign, to each vertex, a y-coordinate proportional to its status.

[2] Best known are KrackPlot [18], Pajek [2], and MultiNet [27]. They mainly offer layouts based on variants of the spring embedder [9], multidimensional scaling, and layouts based on eigenvectors of the adjacency or Laplacian matrix of the graph.

Note how the vertical ordering differs from that in Fig. 2. While the stem-and-leaf diagram of Fig. 3 makes this obvious as well, this visualization also explains the reason why: the measure of status used assigns values according to the values of sending neighbors, and Donna is the only actor that Nancy asks for advice.

While, in principle, any definition of status is applicable, we will see in the next section that, e.g., the requirement of a maximum number of upward pointing edges is not suitable, since it leads to computational difficulties that make interpretation infeasible.

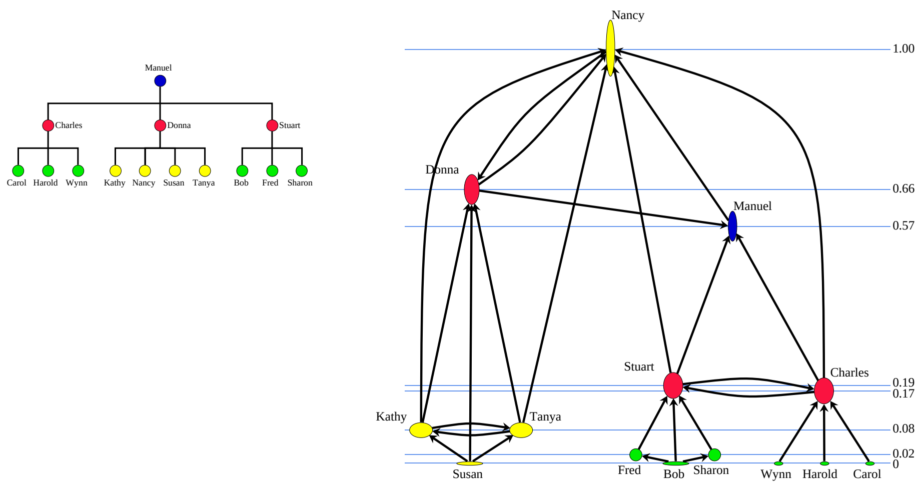


Fig. 4. Contextual visualization of status in advice network (organizational hierarchy shown for comparison)

We have integrated additional information in Fig. 4 by depicting vertices as ellipses rather than circles. This way, the ratio of incoming and outgoing edges is incorporated into the drawing without changing the layout. Let $d_G^-(v)$ and $d_G^+(v)$ denote the in- and outdegree of vertex v. Then, a horizontal radius $r_h(v)$ and a vertical radius $r_v(v)$ for the ellipse are chosen to satisfy

$$\frac{r_v(v)}{r_h(v)} = \frac{d_G^-(v)}{d_G^+(v)},$$
$$r_v(v) + r_h(v) = \pi \cdot d_G^-(v) \cdot d_G^+(v),$$

so that the ratio of in- and outdegrees is visually represented by the ratio of height and width, and the sum of the degrees is represented by the area of a vertex feature. A minimum height and width is used for zero in- and outdegree, and simple adjustments of the second equation account for vertex shapes other than ellipses (rectangles, rhombs).

Other than substantive, there are ergonomic criteria a visualization should satisfy. For example, a large number of crossing edges makes a drawing difficult to read [25]. Visualizations like the one in Fig. 4 are therefore more difficult to produce than, e.g., bar charts, because we can not just place vertices at specified y-coordinates. Algorithms to generate readable drawings under the above substantive constraint are described in the next section.

3 Automatic Layout

To automatically generate layered visualizations of social networks, we have to provide algorithms to compute x-coordinates for vertices and bend points of edges in the graph. This is a special case of a *graph drawing* problem. See [8] for an overview of the field.

The most commonly used framework for horizontally layered drawings of graphs is presented in [28]. It consists of the following generic steps:

1. determine a layer for each vertex,
2. introduce an edge bend point for each layer an edge spans and determine a relative ordering of vertices and bend points on the same layer, and finally
3. assign x- and y-coordinates to each vertex and bend point.

Steps 2 and 3 are separated to enable the use of combinatorial methods in the second step, which serves to reduce the number of crossing edges. Note that crossings severely affect the readability of a drawing [24], and that the number of crossings between two adjacent layers is determined by the relative ordering of vertices and bend points, independent of the actual coordinates (hence the introduction of bend points, see Fig. 5). A comprehensive overview of approaches to carry out the above steps is given in Chapter 9 of [8]. Though there is a whole range of implementations, most notably [11], our specific needs in the first step rule out their usage.

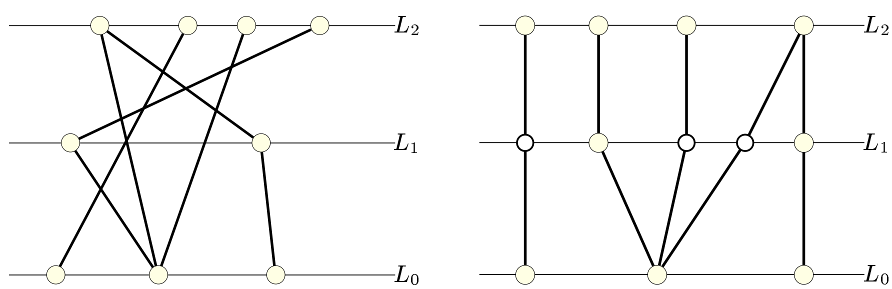


Fig. 5. A three-layer graph with many crossings, and the same graph with reordered vertices and dummy vertices

Layer assignment. We first argue, why the criterion of a maximum number of upward pointing edges must not form the basis of automatically generated status visualizations. A fairly common approach to layering is to break all directed cycles, if any, by temporarily reversing some edges, and assign vertices to layers by topological sorting. Reversing the minimum number of edges nicely corresponds to finding a layering with a maximum number of upward pointing edges.

There are three substantive reasons against this approach. First of all, the implicit definition of status (directed edges imply that the receiver has a higher status than the sender) yields only a partial ordering. Secondly, a minimum cardinality set of cycle breaking edges need not be unique. And thirdly, a straight-forward reduction from feedback arc set shows that the problem of determining such a set with minimum cardinality is $\mathcal{NP}$-hard [15]. Since all three of these difficulties introduce arbitrariness into the complete ordering of actor status that any computed layering implies, interpretation of relative status becomes unreliable, if not impossible.

Assuming that formal status indices have a sound theoretical basis (a discussion of the appropriateness of an interval scale measurement is beyond the scope of this paper), any such index can directly be used for the y-coordinate of each vertex (up to scaling). We don't know of other approaches dealing with y-coordinates that are already given by the context. Let $s = (s_v)_{v \in V}$ be a status vector, a trivial layer assignment then is a partition $L_0 = \{v_0\}, \ldots, L_{|V|-1} = \{v_{|V|-1}\}$ of V, such that $i < j$ implies $s_{v_i} \leq s_{v_j}$. Status values often differ only marginally, though, leading to very close layers that cause perceptual problems like, e.g., several crossing (or non-crossing?) edge segments running almost horizontally (Fig. 6). To avoid such problems, status values are clustered and all vertices with status values in the same cluster are assigned to the same layer. Though any clustering may be used, we apply an agglomerative clustering scheme starting with singletons and merging two clusters, if the minimum status difference between any pair of vertices in different clusters is below a fixed threshold $0 \leq \varepsilon < 1$.

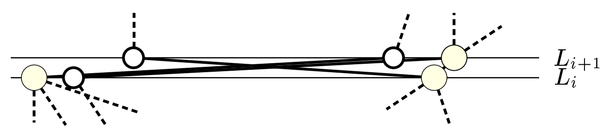


Fig. 6. Readability problems caused by very close layers

Crossing reduction. In this step, we are given a layering $L_0, \ldots, L_k$ of the vertices and our goal is to define a horizontal ordering of vertices in and edges spanning the same layer such that the number of edge crossings is small. An edge $(u, v) \in E$ is said to *span* a layer L_i, if $u \in L_{j_1}$, $v \in L_{j_2}$, and $j_1 < i < j_2$ or $j_2 < i < j_1$. For each layer an edge spans, a dummy vertex representing a bend point is introduced, subdividing that edge and placed in the appropriate layer. We can now assume that we are given a layering such that no edge spans any layer. Note that the number of crossings is now dependent only on the ordering of vertices in each layer.

Finding an ordering that minimizes the number of edge crossings is another $\mathcal{NP}$-hard problem [12]. A common heuristic is the layer-by-layer sweep, in which the ordering in, say, L_0 is fixed and L_1 is reordered to reduce the number of crossings. Then, the order in L_1 is fixed, and L_2 is reordered, and so on. After reaching L_k, the process is reversed and repeated up and down the layering until no further improvement is made. Note however, that minimizing the number of crossings between adjacent layers, where the ordering in one layer is fixed, is $\mathcal{NP}$-hard [10] as well. Though in praxis this problem can be solved optimally for medium sized instances [14] using integer linear programming, the overall number of crossing will not be minimum. For simplicity, we use one of several heuristics (e.g., the median heuristic, placing a vertex at the median position of its neighbors in the adjacent layer) which are known to perform quite satisfactory.

Another heuristic, called global sifting [20], is used as a postprocessing step to the layer-by-layer sweep to reduce the number of remaining crossing. Roughly speaking, global sifting picks one vertex at a time and finds the locally optimal position within a layer by probing all of them. Our experiences are that this postprocessing is worth the additional effort.

Horizontal placement. Given y-coordinates, a layering, and an ordering of vertices and bend points within each layer, it remains to compute x-coordinates respecting the horizontal orderings. Currently, we are using an adoption of a fast heuristic provided in the AGD library [23], trying to straighten long edges and keep edge lengths small, but better strategies need to be explored for future use.

4 Example and Conclusion

One of several studies already applying our visualization approach is an analysis of the privatization processes of two industrial conglomerates in Eastern Germany after reunification [26]. Actors in these networks are political or corporate organizations, and different kinds of relations between them are investigated.

In this application, we represent the semantic attributes “sector” (government, political parties, unions and associations, corporations) by color and “level” (local, regional, federal) by shape. To reduce clutter due to bidirectional edges and arrow heads, non-downward pointing uni-directional edges are depicted in black, bidirectional edges in green, and downward pointing edges in red.

Figure 7 shows two relations between the same set of actors in the ship-building industry. On the left, edges indicate to what other actors an actor reports mandatorily, and on the right, edges indicate whose interests actors claim to have taken into account in important decisions. Even without any background knowledge, it is readily observed that fairly coordinated high-level governmental actors (blue rhombs) dominate the hierarchy of interest consideration.

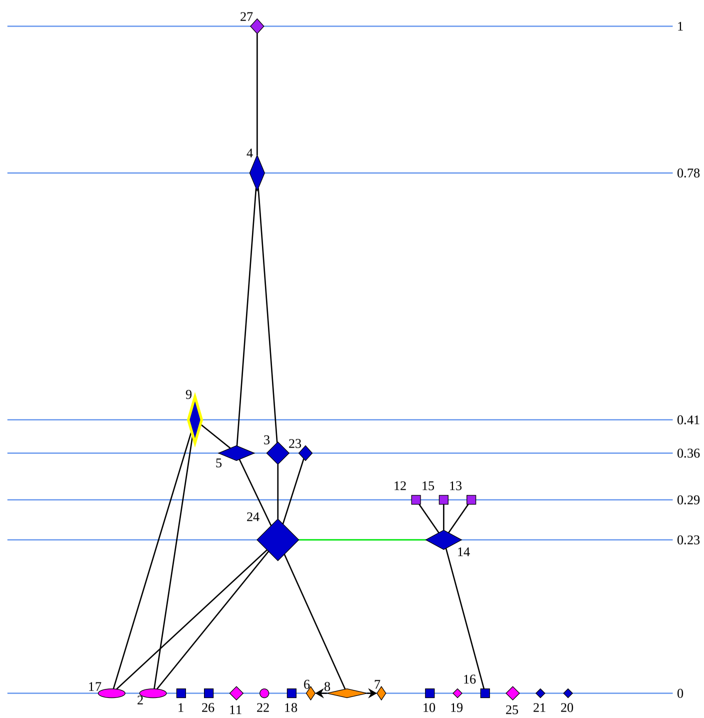
Though our visualizations are considered very useful by those using them, we feel that several details – in particular regarding bend point placement – need further improvement. Moreover, we would like to provide automatic help for label placement, which has been refined manually for the above examples, and need to explore means of user interaction: what kind of improvements may a user make without running the risk of being suggestive? Similar work [6] is concerned with a structural index called centrality, but can we also provide automatic support for contextual visualizations of substance without an immediate geometric connotation?

Acknowledgments. Data sets are courtesy of David Krackhardt and Jörg Raab. Frank Müller implemented a prototype layout system in C++ using LEDA [22], AGD [23], and LAPACK [1]. This work would not have been possible without the fruitful discussions we had with Patrick Kenis, Jörg Raab, and Volker Schneider.

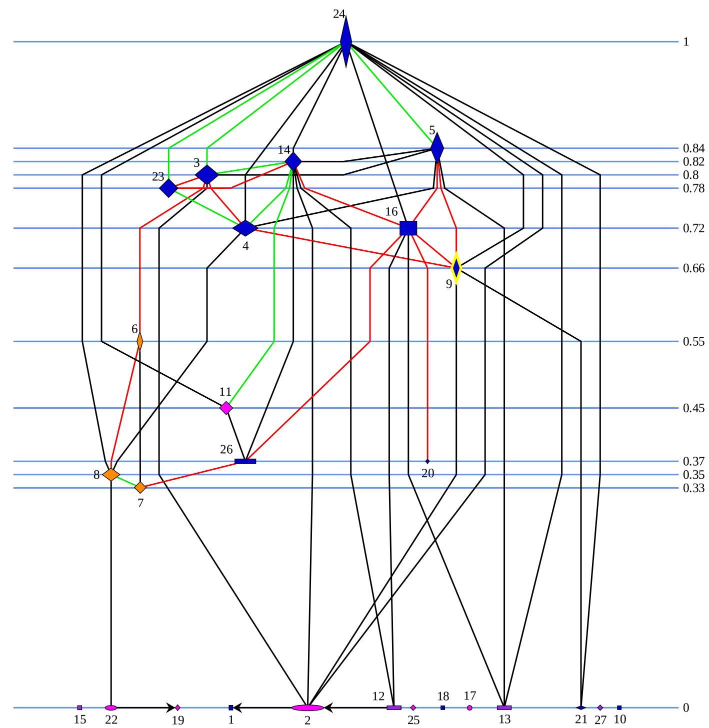
References

1. E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK User's Guide*. Society for Industrial and Applied Mathematics, 3rd edition, 1999. See `http://www.netlib.org/lapack/`.
2. V. Batagelj and A. Mrvar. PAJEK – Program for large network analysis. *Connections*, 21(2):47–57, 1998. Project home page at `http://vlado.mat.uni-lj.si/pub/networks/pajek/default.htm`.
3. R. A. Becker, S. G. Eick, and A. R. Wilks. Visualizing network data. *IEEE Transactions on Visualization and Graphics*, 1(1):16–28, 1995.
4. J. Bertin. *Semiology of Graphics: Diagrams, Networks, Maps*. University of Wisconsin Press, 1983.
5. U. Brandes, P. Kenis, J. Raab, V. Schneider, and D. Wagner. Explorations into the visualization of policy networks. *Journal of Theoretical Politics*, 11(1):75–106, 1999.
6. U. Brandes, P. Kenis, and D. Wagner. Centrality in policy network drawings. In Kratochvíl [19], pages 250–258.
7. R. S. Burt. *Structure, Version 4.2*. Center for the Social Sciences, Columbia University, New York, 1991.
8. G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1999.
9. P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160, 1984.
10. P. Eades and N. C. Wormald. Edge crossings in drawings of bipartite graphs. *Algorithmica*, 11:379–403, 1994.
11. E. R. Gansner, S. C. North, and K.-P. Vo. DAG – A program that draws directed graphs. *Software—Practice and Experience*, 17(1):1047–1062, 1988.
12. M. R. Garey and D. S. Johnson. Crossing number is $\mathcal{NP}$-complete. *SIAM Journal on Algebraic and Discrete Methods*, 4:312–316, 1983.
13. N. B. Harrison and J. O. Coplien. Patterns of productive software organziations. *Bell Labs Technical Journal*, pages 138–145, Summer 1996.
14. M. Jünger and P. Mutzel. 2-Layer straightline crossing minimization: Performance of exact and heuristic algorithms. *Journal on Graph Algorithms and Applications*, 1(1):1–25, 1997.
15. R. M. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.
16. L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18:39–43, 1953.
17. D. Krackhardt. Social networks and the liability of newness for managers. In C. L. Cooper and D. M. Rousseau, editors, *Trends in Organizational Behavior*, volume 3, pages 159–173. John Wiley & Sons, 1996.
18. D. Krackhardt, J. Blythe, and C. McGrath. KrackPlot 3.0: An improved network drawing program. *Connections*, 17(2):53–55, 1994. Project home page at `http://www.contrib.andrew.cmu.edu/~krack/`.
19. J. Kratochvíl, editor. *Proceedings of the 7th International Symposium on Graph Drawing (GD '99)*, volume 1731 of *Lecture Notes in Computer Science*. Springer, 1999.

20. C. Matuszewski, R. Schönfeld, and P. Molitor. Using sifting for k-layer straightline crossing minimization. In Kratochvíl [19], pages 217–224.
21. C. McGrath, J. Blythe, and D. Krackhardt. The effect of spatial arrangement on judgments and errors in interpreting graphs. *Social Networks*, 19(3):223–242, 1997.
22. K. Mehlhorn and S. Näher. *The LEDA Platform of Combinatorial and Geometric Computing*. Cambridge University Press, 1999. Project home page at `http://www.mpi-sb.mpg.de/LEDA/`.
23. P. Mutzel, C. Gutwenger, R. Brockenauer, S. Fialko, G. W. Klau, M. Krüger, T. Ziegler, S. Näher, D. Alberts, D. Ambras, G. Koch, M. Jünger, C. Buchheim, and S. Leipert. A library of algorithms for graph drawing. In S. H. Whitesides, editor, *Proceedings of the 6th International Symposium on Graph Drawing (GD '98)*, volume 1547 of *Lecture Notes in Computer Science*, pages 456–457. Springer, 1998. Project home page at `http://www.mpi-sb.mpg.de/AGD/`.
24. H. C. Purchase. Which aesthetic has the greatest effect on human understanding? In G. Di Battista, editor, *Proceedings of the 5th International Symposium on Graph Drawing (GD '97)*, volume 1353 of *Lecture Notes in Computer Science*, pages 248–261. Springer, 1997.
25. H. C. Purchase, R. F. Cohen, and M. James. An experimental study of the basis for graph drawing algorithms. *ACM Journal of Experimental Algorithmics*, 2(4), 1997.
26. J. Raab. *Steuerungsstrukturen politisierter Großprivatisierungen in Ostdeutschland 1990–1994. Das Beispiel der Werft- und Stahlindustrie*. PhD thesis, University of Konstanz, 2000.
27. W. D. Richards. MultiNet. Software tool, see `http://www.sfu.ca/~richards/Multinet/`.
28. 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, February 1981.
29. S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994.



(a) mandatory reporting



(b) interest consideration

Fig. 7. Status according to a measure implemented in Structure [7] in two networks with the same set of political organizations