visone
Analysis and Visualization of Social Networks*

Ulrik Brandes[1] and Dorothea Wagner[2]

[1] University of Passau, Department of Mathematics & Computer Science, 94030 Passau, Germany. `brandes@algo.fmi.uni-passau.de`

[2] University of Konstanz, Department of Computer & Information Science, 78457 Konstanz, Germany. `Dorothea.Wagner@uni-konstanz.de`

1 Introduction

We describe visone, a tool that facilitates the visual exploration of social networks. Social network analysis is a methodological approach in the social sciences using graph-theoretic concepts to describe, understand and explain social structure. The visone software is an attempt to integrate analysis and visualization of social networks and is intended to be used in research and teaching. While we are primarily focussing on users in the social sciences, several features provided in the tool will be useful in other fields as well.

In contrast to more conventional mathematical software in the social sciences that aim at providing a comprehensive suite of analytical options, our emphasis is on complementing every option we provide with tailored means of graphical interaction. We attempt to make complicated types of analysis and data handling transparent, intuitive, and more readily accessible. User feedback indicates that many who usually regard data exploration and analysis complicated and unnerving enjoy the playful nature of visual interaction.

Consequently, much of the tool is about graph drawing methods specifically adapted to facilitate visual data exploration. The origins of visone lie in an interdisciplinary cooperation with researchers from political science which resulted in innovative uses of graph drawing methods for social network visualization, and prototypical implementations thereof. With the growing demand for access to these methods, we started implementing an integrated tool for public use. It should be stressed, however, that visone remains a research platform and testbed for innovative methods, and is not intended to become

* Many people have contributed directly or indirectly to the current state of our tool. We thank Sabine Cornelsen, Patrick Kenis, Jörg Raab, and Volker Schneider for many years of fruitful cooperation, and the participants of POLNET summer schools for their feedback and suggestions. We are indebted to Michael Baur, Marc Benkert, Marco Gaertler, Boris Köpf, and Jürgen Lerner for their implementation efforts, and gratefully acknowledge financial support from the Deutsche Forschungsgemeinschaft (DFG) under grant BR 2158/1-1 and the European Commission within FET Open Project COSIN (IST-2001-33555).
© visone logos by Christiane Nöstlinger and Ulrik Brandes.

a standard tool with all due consequences such as extensive user-support and product marketing. Essentially all components are in development and therefore subject to change. In a nutshell, visone is a

- tool for interactive analysis and visualization of networks, in which
- originality is preferred over comprehensiveness, and that
- caters especially to social scientists.

The organization of the subsequent sections follows the common structure of all chapters in this book. In particular, we start with background information on the main area of application for visone, and give application examples in Section 5. While other interesting algorithms have been implemented, Section 3 focusses on those for graph drawing.

2 Applications

The main application area of visone is a methodological approach in the social sciences: *Social Network Analysis* uses graph-theoretic concepts to describe, understand and explain, sometimes even predict or design, social structure. The objects of interest are emergent patterns of relationships and their interplay with entity attributes.

To motivate the decisions made in the design of visone, we describe the data model on which we operate, types of analysis provided, and visualization principles governing our choice of graph drawing algorithms.

2.1 Model

A social network consists of *nodes* (often referred to as *actors*), i.e. entities such as persons, organizations, or simply objects that are *linked* by binary relations such as social relations, dependencies, or exchange. Both nodes and links may have additional *attributes*.

Relations constituting a social network may be directed, undirected, or mixed. Attributes can be of any type, and numerical link attributes may strengthen or weaken the tie between two nodes. Since data is often gathered by means of questionnaires, even the existence of a link is subject to interpretation because two respondents may have different perceptions regarding the presence of a specific type of tie between them, i.e. the link may be confirmed or unconfirmed. Rather typical examples of the kind of structures studied are given in Section 5.

To simplify usage, implementation, and documentation, visone operates on a single, unified network model (in essence, a labeled digraph) that is general enough to capture the essential features of a broad range of conceivable cases. Since the tool is interactive, objects may be selected to alter the subgraph to which an operation is applied. The rules on how particular network features are mapped to the uniform model are described in Section 6.

Definition 1 (visone network model). A *social network* is a labeled directed graph $G = (V, E = E_C \cup E_U; \delta, \omega)$,[1] where E_C and E_U are disjoint sets of *confirmed* and *unconfirmed* edges, $\delta : E \to \mathbb{R}_{\geq 0}$ is a non-negative edge *length*, and $\omega : E \to \mathbb{R}_{\geq 0}$ a non-negative edge *strength*.

A vertex or edge *attribute* is a (partial) function assigning values to vertices or edges. The values assigned by a *nominal* attribute are strings, while those of a *numerical* attribute are non-negative real numbers.

A crucial feature in many studies is the interrelation between structural properties of a social network and its attributes. We hence provide convenient mechanisms to handle an arbitrary number of vertex and edge attributes. These can be mapped to the visual appearance of the graph, or used to define length and strength labels and thus influence the outcome of an analysis.

Although there is no restriction on the class of graphs that constitute a social network, instances from social science projects tend to be sparse but locally dense, and to exhibit small average distances between vertices. Moreover, these graphs are frequently small to medium in size. We thus assume that, roughly, $n + m \leq 1000$ and consider algorithms running in time $\mathcal{O}(nm)$ (for reasonable constants) to be acceptable. For significantly larger graphs, we recommend to try `Pajek`, a tool for the analysis of large networks also described in this book.

2.2 Analysis

The purpose of social network analysis is to identify important actors, crucial links, subgroups, roles, network characteristics, and so on, to answer substantive questions about structures.

There are three main levels of interest: the element, group, and network level. On the element level, one is interested in properties (both absolute and relative) of single actors, links, or incidences. Examples for this type of analyses are bottleneck identification and structural ranking of network items. On the group level, one is interested in classifying the elements of a network and properties of subnetworks. Examples are actor equivalence classes and cluster identification. Finally, on the network level, one is interested in properties of the overall network such as connectivity or balance.

Currently, the types of analyses provided in visone are almost exclusively on the element level (with corresponding network level statistics). More specifically, we have focussed on indices measuring structural importance of vertices. While there is no universally accepted definition of what makes a vertex important, a small collection of indices forms the basis of most studies. Several of these originally do not apply to our rather general network model (e.g., some only apply to connected undirected graphs), but we were able

[1] Recall that the definition in Section 2 of the Technical Foundations Chapter allows for multiple edges and self-loops.

to generalize and unify them. The complete list of currently implemented indices is given in Figure 1.

index	**definition**	**reference**
	local measures	
degree	$c_v = \sum_{e \in \mathrm{instar}(v) \cup \mathrm{outstar}(v)} \omega(e)$	–
indegree	$c_v = \sum_{e \in \mathrm{instar}(v)} \omega(e)$	–
outdegree	$c_v = \sum_{e \in \mathrm{outstar}(v)} \omega(e)$	–
	distance measures	
betweenness	$c_v = \sum_{s \neq v \neq t \in V} \frac{\sigma_G(s,t\|v)}{\sigma_G(s,t)}$ where $\sigma_G(s,t)$ and $\sigma_G(s,t\|v)$ are the number of all shortest st-paths and those passing through v	[2,19,9]
closeness	$c_v = \frac{1}{\sum_{t \in V} \delta(v,t)}$	[7,32]
eccentricity	$c_v = \frac{1}{\max_{t \in V} \delta(v,t)}$	[21]
radiality	$c_v = \frac{\sum_{t \in V} (\mathrm{diam}(G) + 1 - \delta(v,t))}{(n-1) \cdot \mathrm{diam}(G)}$	[35]
	feedback measures	
status	$c_v = \alpha \cdot \sum_{(u,v) \in \mathrm{instar}(v)} (1 + c_u)$ where $\alpha = \min\{\max_{v \in V} \mathrm{indeg}(v), \max_{v \in V} \mathrm{outdeg}(v)\}^{-1}$	[24]
eigenvector	$c_v = \mu^{-1} \sum_{(u,v) \in \mathrm{instar}(v)} \omega(u,v) \cdot c_u$ where μ is the largest eigenvalue of $A(G)$	[8]
pagerank	$c_v = \gamma \cdot \frac{1}{n} + (1-\gamma) \sum_{(u,v) \in \mathrm{instar}(v)} c_u$ where $0 < \gamma < 1$ is a free parameter	[15]
authority	$c_v = \mu^{-1} \cdot \sum_{(u,v) \in \mathrm{instar}(v)} \omega(u,v) \cdot \sum_{(u,w) \in \mathrm{outstar}(u)} \omega(u,w) c_w$ where μ is the largest eigenvalue of $A(G)^T A(G)$	[26]
hub	$c_v = \mu^{-1} \cdot \sum_{(v,w) \in \mathrm{outstar}(v)} \omega(v,w) \cdot \sum_{(u,w) \in \mathrm{instar}(w)} \omega(u,w) c_u$ where μ is the largest eigenvalue of $A(G) A(G)^T$	[26]

Fig. 1. Available vertex centralities. Note that most indices have been generalized with respect to the original references, and all are rescaled to percentages.

One particular consequence of our unification is that all vertex indices are non-negative and have unit sums, i.e. they can be viewed as probability distributions on the vertex set and interpreted as the share of importance a node assumes in its network.

Since the theory for edge indices is even less developed, we are currently investigating extensions to edges along the same lines. Moreover, support for graphic comparison of different vertex or edge indices and for some types of group level analysis is intended to be added in the future, but will require entirely different forms of visualization (cf. next subsection). A comprehensive, though non-visual, tool for social network analysis is UCINET [1].

Note that it is a long-standing debate whether unconfirmed edges should be considered for analysis. Typically, researchers decide to either treat all unconfirmed edges/indexedge!unconfirmed as if confirmed, or to exclude them completely. We leave this decision with the researcher, but add the freedom to make it on a per-edge basis (cf. Section 6). This way, the user has full control over the assumptions made, and the ability to experiment with different hypotheses and compare their consequences.

2.3 Visualization

Visualized information must neither be misleading nor hard to read. Hence there are two obvious criteria for the quality of social network visualizations:

1. Is the information manifest in the network represented accurately?
2. Is this information conveyed efficiently?

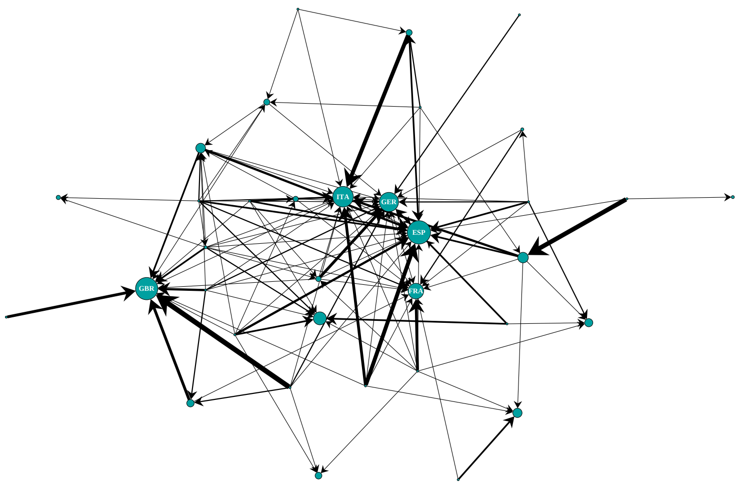
With these criteria in mind, the following three aspects should be carefully thought through when creating network visualizations [11]:

- the *substantive aspect* the viewer is interested in,
- the *design* (i.e. the mapping of data to graphical variables), and
- the *algorithm* employed to realize the design (artifacts, efficiency, etc.).

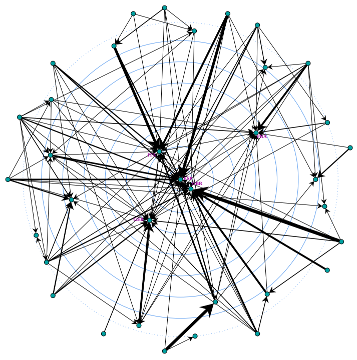
In addition to algorithms that try to produce what is often termed an "aesthetic" drawing of a graph (and thus are oblivious to the first aspect) we developed the following two types of visualization specifically for the vertex index analyses currently available in visone.

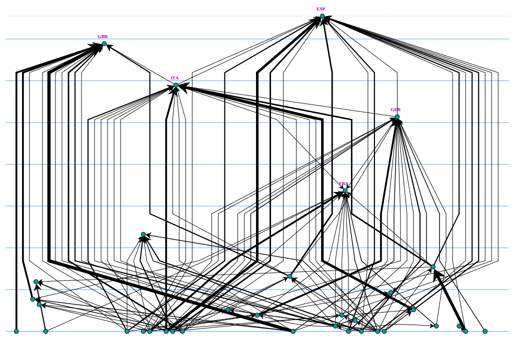
Depending on the context, actors of high structural importance are interpreted as a being *central* or as having *high status*. With this substantive aspect in mind, we designed visualizations that represent vertex indices by constraining vertex positions to fixed distances from the center or from the bottom of the drawing, in either case depending linearly on the vertex index. See Figure 2 for illustration and note that relative scores are difficult to determine from the straightforward representation based on vertex size.

The information can thus be represented accurately, and it is up to the (constrained) graph layout algorithm to optimize readability. To avoid user



(a) vertex index represented by vertex size





(b) interpreted as centrality (c) interpreted as status

Fig. 2. Different means of visualizing a vertex index: most prestigious football leagues based on which ones the participants of the 1998 World Cup Final played in (network data courtesy of Lothar Krempel). Thickness of edges indicates number of players in foreign league. Like graph paper, background lines support determination and comparison of scores.

dissatisfaction with suboptimal drawings, we strive to find at least locally optimal layouts that are not obvious for users to improve. The algorithms used for status and centrality drawings are described in Sections 3.3 and 3.4.

3 Algorithms

From the computer science point of view, one of the main aspects of the visone project and software is that of a stimulus and testbed for algorithmic research. Indeed, new and more efficient algorithms have been developed for many components of the tool.

For example, more efficient generators have been implemented to create graphs according to popular stochastic models such as random graphs [20], small worlds [36], and evolving graphs with preferential attachment [3]. Time and space complexity of these generators is linear in the size of the graph generated [4].

For vertex indices, not only unified definitions and normalizations, but also unified algorithms are introduced in visone. While all feedback measures are computed using variants of sparse-matrix power iteration, all distance measures are determined by solving an augmented single-source shortest path problem from each vertex. For betweenness centrality, in particular, this yields a substantial improvement over previous algorithms [9].

Though we are facing many more interesting algorithmic challenges during the course of this project, we focus here on some that arise in the context of visualization, the main topic of this book. In the subsections below we describe our approaches for the more involved types of layouts provided.

3.1 Uniform layouts

When exploring a network, spring embedder layouts are useful to catch a first glimpse of the overall structure of the graph. However, the algorithms' performance and layout quality tend to worsen significantly with increasing size of the graph.

A related, yet more reliable approach to draw very large graphs is introduced in [23], but limited to undirected graphs without tree-like substructures. We first describe the original approach and then sketch extensions we are currently implementing to take edge directions into account and alleviate the problems caused by low connectivity.

A high-dimensional embedding approach. Let $G = (V, E)$ be a simple and connected undirected graph with vertices $V = \{v_1, \ldots, v_n\}$. Similar to the spring embedder variant of Kamada and Kawai, the basic goal is to place every pair of vertices at a distance proportional to its graph-theoretic distance. Rather than placing vertices directly in two dimensions, first an n-dimensional layout is determined in which each dimension is contingent on a different vertex. In the dimension of a vertex $v \in V$, the coordinate of each vertex is its centered distance from v in the graph, $p_v^{(i)} = d_G(v, v_i) - \frac{1}{n} \sum_{w \in V} d_G(w, vi)$. This high-dimensional drawing is projected down into two dimensions using principal component analysis. That is, a projection with maximum variance is determined from two eigenvectors associated with the largest eigenvalues of the covariance matrix $\Sigma = (\sigma_{ij})_{1 \leq i,j \leq n}$, where

$$\sigma_{ij} = {p^{(i)}}^T \cdot p^{(j)} \ .$$

With eigenvectors $e^{(1)}, e^{(2)}$, which can be computed using power iteration, the location $p_v = (x_v, y_v)$ of $v \in V$ is obtained from

$$x_v = \sum_{i=1}^{n} e_i^{(1)} \cdot p_v^{(i)} \quad \text{and} \quad y_v = \sum_{i=1}^{n} e_i^{(2)} \cdot p_v^{(i)} \ .$$

Note that, because of the size of the covariance matrix, the overall algorithm has running time $\Omega(d^2 n)$, where d is the number of dimensions of the high-dimensional embedding. Thus, if the number n of vertices is large, only a sample is used to determine the initial embedding. A simple heuristic for the k-center problem serves well to select that sample [23].

Modifications. The above approach cannot take into account the direction of edges. Likewise, it is not suitable for large graphs of low connectivity. Consider the block-cutpoint tree of a non-biconnected graph. If a subtree contains none of the sample vertices, all vertices in the subtree will be placed at the same relative positions in every dimension, and thus in the projection.

We are therefore modifying the high-dimensional embedding approach in several ways: mainly, we reserve some of the dimensions of the initial embedding to display edge directions, and introduce dependencies between others to avoid strong correlations between substructure layouts. We also consider edge lengths in the computation of distances. Finally, we add dimensions in which only the subgraph induced by confirmed edges is considered to make it visually more dominant.

3.2 Spectral layouts

Let $G = (V, E; \omega)$ be an undirected graph with positive edge weights, e.g. obtained from the underlying undirected graph of a social network and its strength label ω. Consider the following weighted version of the minimization objective of Tutte's barycentric layout model (cf. Section 4.5 of the Technical Foundations Chapter)

$$\sum_{\{v,w\} \in E} \omega(e) \cdot \|p_v - p_w\|^2 = \sum_{\{v,w\} \in E} \omega(e) \cdot \left((x_v - x_w)^2 + (y_v - y_w)^2\right) \qquad (1)$$

where $p_v = (x_v, y_y) \in \mathrm{I\!R}^2$ is the location of vertex $v \in V$. Recall that optimum solutions place all vertices in the same location. In *spectral graph layout*, first introduced by Hall [22], these undesirable solutions are avoided not by fixing the location of select vertices, but by putting more uniform constraints on the location vector $p = (p_v)_v \in V$ as follows.

In matrix notation, Tutte's objective (1) can be expressed as $p^T L(G)p$, where

$$L(G) = D(G) - A(G)$$

is called the *Laplacian matrix* of G, with $D(G)$ the diagonal matrix of vertex degrees and $A(G)$ the weighted adjacency matrix. To eliminate the dependency on the scale of p we divide this quadratic form by $p^T p = \|p\|^2$. Now observe that, if p is an eigenvector of $L(G)$, the associated eigenvalue is $\frac{p^T L(G)p}{p^T p}$, and that the trivial optima of (1) are multiples of $p = \mathbf{1}$, i.e. the vector with all components equal to one, and associated with eigenvalue 0.

The eigenvalues of the Laplacian are non-negative real numbers, and their eigenvectors are pairwise orthogonal. Two eigenvectors associated with the smallest non-zero eigenvalues of $L(G)$ therefore minimize

$$\sum_{\{v,w\}\in E} \omega(e) \cdot (x_v - x_w)^2 = \frac{x^T L(G)x}{x^T x} \qquad \text{subject to } \mathbf{0} \neq x \perp \mathbf{1}$$

and

$$\sum_{\{v,w\}\in E} \omega(e) \cdot (y_v - y_w)^2 = \frac{y^T L(G)y}{y^T y} \qquad \text{subject to } \mathbf{0} \neq y \perp \mathbf{1} \text{ and } y \perp x\ .$$

As a consequence of orthogonalization with $\mathbf{1}$, the resulting layouts are centered around the origin.

Symmetries are displayed well in spectral layouts, and structurally equivalent vertices (i.e. vertices with identical neighborhoods) are placed in the same location. If a graph is not balanced, however, most vertices are clustered in the center of the drawing, and only some loosely connected vertices are placed far away. To counter this effect, a slightly modified Laplacian $L_\rho(G) = (1 - \rho)D(G) - A(G)$ in which the diagonal is weakened by a constant factor ρ, $0 \leq \rho \leq 1$, is used. This can be viewed as pushing vertices out of the center by applying a radial force that depends on the degree of a vertex and is illustrated in Figure 3.

Since the graphs we deal with are of medium size, no sophisticated algorithm is needed. The eigenvalues are simply reversed using an upper bound Λ on the largest one, so that power iteration with re-orthogonalization yields the two desired eigenvectors. The current positions on the screen are used for initialization, and the *residual*, i.e. the squared distance of a vector from being an eigenvector, serves as convergence criterion. The entire layout algorithm is given in Alg. 1. Note that $\rho = 0$ yields the standard spectral layout, whereas $\rho = 1$ yields two eigenvectors of the adjacency matrix.

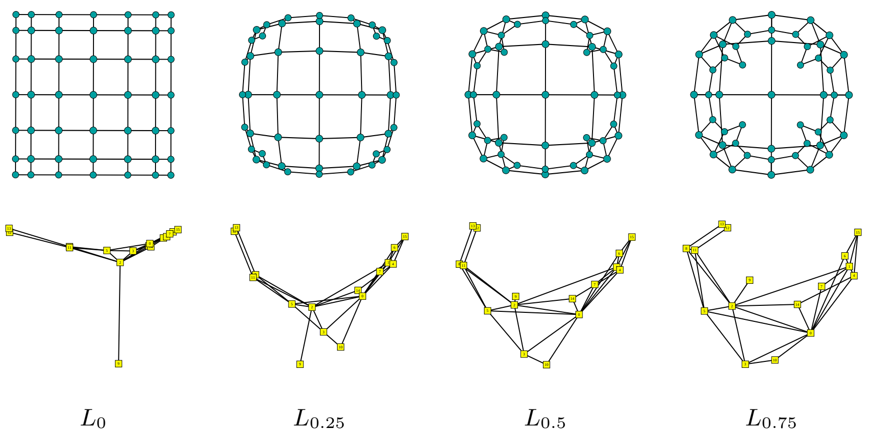


Fig. 3. Spectral layouts with modified Laplacian L_ρ. The graph in the lower row is from Figure 21 of the Technical Foundations Chapter.

Algorithm 1: Spectral layout with a modified Laplacian matrix

```
**Input**: undirected graph $G = (V, E; \omega)$ with edge strengths, $n = |V|$
           initial layout $p_v = (x_v, y_v)$, $v \in V$
           parameter $0 \leq \rho \leq 1$
**Output**: layout $p_v = (x_v, y_v)$, $v \in V$

$r \leftarrow 2$
**while** $r > 1$ **do**
    $x' \leftarrow (\Lambda \cdot I - L_\rho(G)) \cdot x$;   $y' \leftarrow (\Lambda \cdot I - L_\rho(G)) \cdot y$
    $x' \leftarrow x' - (\sum_{v \in V} \frac{x'_v}{n}) \cdot \mathbf{1}$;   $y' \leftarrow y' - (\sum_{v \in V} \frac{y'_v}{n}) \cdot \mathbf{1}$
    $y' \leftarrow y' - \frac{y'^T x'}{x'^T x'}$
    $r \leftarrow \max\{\|x' - \frac{x^T x'}{x^T x} x\|^2, \|y' - \frac{y^T y'}{y^T y} y\|^2\}$
    $x \leftarrow \frac{n}{\max_{v \in V} x'_v} x'$;   $y \leftarrow \frac{n}{\max_{v \in V} y'_v} y'$
**od**
```

With an additional fixed upper bound on the number of iterations, Alg. 1 runs in time $\mathcal{O}(n + m)$. Since the current positions on the screen are used for initialization, more iterations can be performed simply by calling spectral layout again. Note that, using more elaborate multi-scale methods, spectral layouts can be computed efficiently even for very large graphs [27].

3.3 Layered layouts

To visually support status analyses of networks as described in Section 2, an algorithm for layered graph layouts is provided. The algorithm is a particular instance of the Sugiyama framework (cf. Section 4.2 of the Technical

Foundations Chapter), with some rather unusual modifications induced by our special setting.

In particular, our layered layout algorithm does not modify relative vertical positions of vertices. This is because our visualization criteria demand that the vertex index be represented precisely. The purpose of the layout algorithm is therefore to make the drawing as readable as possible without changing y-coordinates. The algorithm is described along the three main phases of the Sugiyama framework and a refinement of what is outlined in [14]. Note that we treat each connected component separately.

Layer assignment. Fixed y-coordinates immediately induce a layering in which vertices with equal vertical position are placed in the same layer. However, typical status indices then yield layerings with many singleton layers and pairs of layers with tiny vertical distance so that edges running between them are almost indistinguishable.

Instead, we run a one-dimensional clustering algorithm on the set of y-coordinates assumed by vertices, and treat each cluster as a layer.

Crossing minimization. It is particularly difficult for crossing reduction procedures to untangle sparsely connected subgraphs. Since we strive for layouts that appear difficult to improve, we start by removing all dangling trees. Note that a layered tree is trivially ordered to have no crossing.

For crossing reduction we apply the barycenter heuristic, followed by a weighted variant of sifting [29]. While the barycenter heuristic is fast and good at separating biconnected components, a few rounds of subsequent sifting ensure that we end up with a layout that cannot be improved by moving a single vertex.

In our weighted variant, each crossing contributes the product of the two edge weights involved, where edges are weighted according to their thickness on the screen. Recall that a social network has two types of edges, confirmed and unconfirmed. Confirmed edges are considered to be more important, and it should be possible to recognize the subgraph induced by confirmed edges in the drawing of the overall graph. To discourage crossings between confirmed edges, their weight is doubled in the algorithm.

After vertex orderings have been determined for the reduced components, the temporarily removed dangling trees are re-inserted into the ordering.

Finally, we make sure that pairs of long edges do not cross at inner segments, so that they can be drawn with only two bends (at their extreme dummy vertices). Note that crossings can be moved up or down by swapping the order of dummy vertices on one layer. We move crossings downward, because the more important part of the drawing is the top – i.e. where the high status actors are (cf. Figure 2).

Coordinate assignment. Using the linear-time algorithm of [13] we obtain integer horizontal coordinates that are subsequently scaled to fit the entire graph on the screen.

3.4 Radial layouts

We provide an algorithm for radial graph layouts to support centrality visualizations as described in Section 2.3.

A radial layout is described in polar coordinates $p_v = (r_v, \varphi_v)$, $v \in V$, but since we use them to convey a structural vertex centrality index c, the first coordinate of vertices $v \in V$ with $c_v > 0$ is fixed at $r_v = \frac{c_v - \underline{c}}{\overline{c} - \underline{c}}$, where $\overline{c}$ and $\underline{c}$ are the maximum and the minimum non-zero score. If the two highest centrality scores differ only marginally, the range of radii is reduced by a fixed offset to avoid vertex overlap in the center. Vertices with zero centrality are placed on an outer orbit.

Similar to computing x-coordinates in layered layouts, the angular φ-coordinates are determined so as to increase the readability of the diagram. The main layout objectives are uniform distribution of vertices, and few edge crossings. While the three-stage force-directed method of [12] yields the most appealing layouts to date, it is too slow for an interactive tool. A faster, purely combinatorial algorithm is therefore used.

Note that, different from the layered case, the (cyclic) ordering of vertices in (circular) layers does not even determine the number of edge crossings. Instead of the radial layout problem we therefore restrict our attention to *circular layouts*, i.e. to the case $r_v = r_w > 0$ for all $v, w \in V$. Nevertheless, crossing minimization is $\mathcal{NP}$-hard even for circular layouts [28].

Similar to the approach taken for layered layouts in the previous subsection, we split the graph into its connected components and treat them individually. In fact, it is split into its biconnected components, because their layouts can be combined without introducing any additional crossings (see Figure 4).

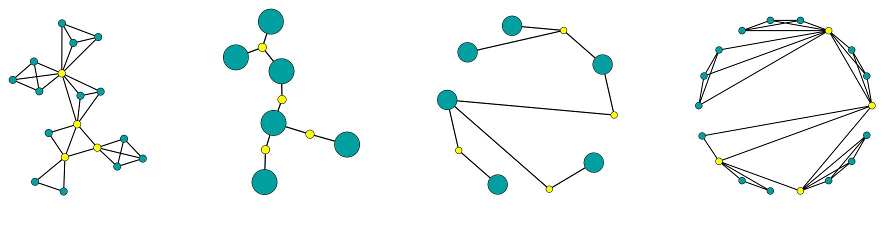


Fig. 4. Utilizing the tree of blocks and cut-vertices to avoid crossings between edges that belong to different blocks.

An efficient algorithm for circular layouts of biconnected graphs is introduced in [34,33] and experimentally shown to produce fewer crossings than previous approaches. A simpler algorithm [6] based on circular sifting yields even fewer crossings. Starting from the cyclic ordering given by current positions on the screen, the idea is to iteratively place a single vertex in its locally optimal position, i.e. where the number of crossings in which its incident edges are involved is minimized. To find this position, the vertex is moved around the circle, and each time the change in the number of crossings is recorded.

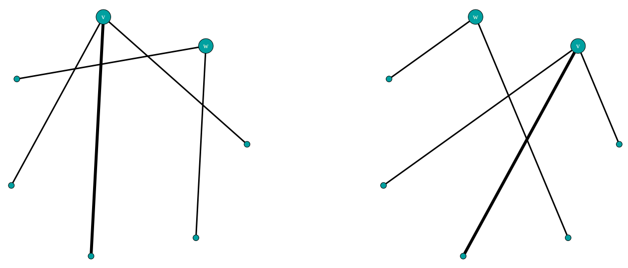


Fig. 5. After swapping two neighboring vertices v and w, an edge of v crosses exactly those edges of w that it didn't cross before.

Assume that all adjacency lists are cyclic and ordered according to the current cyclic ordering of vertices. This can be achieved using bucket sort in time $\mathcal{O}(m)$. The vertex v to be placed optimally is moved clockwise, one position at a time. When swapping the vertex with a neighbor w, the resulting difference in the number of crossings is determined by merging the two sorted adjacency lists. For each edge incident to v, the change in crossings is the difference between the length of the prefix and suffix of the current position in the adjacency list of w. See Figure 5 for an illustration.

A single swap takes time $\mathcal{O}(\deg(v)+\deg(w))$. Locally optimal positioning once for each vertex therefore takes amortized time $\mathcal{O}(nm)$. Experimental evidence indicates that a few such rounds suffice, and that this algorithm consistently outperforms other heuristics.

4 Implementation

The visone software is implemented in C++ using LEDA, the *Library of Efficient Data Types and Algorithms* [30]. While the user interface is a customized version of LEDA's GraphWin class, all graph generation, analysis, and layout algorithms (except for LEDA's force-directed layout routine) have been implemented from scratch.

Starting with version 1.1, the main data format used in visone will be the XML sublanguage GraphML (Graph Markup Language) [10]. GraphML support is implemented in a LEDA extension package which will be made available for public use. It will hence be possible to administer project files with several social networks and any number of attributes. Figure 6 shows a self-explanatory fragment of a social network represented in GraphML. Data attributes can be mapped freely to graphical attributes like color, shape, and so on.

```
...
<key id="k0" for="edge"
     attr.name="visone:confirmed" attr.type="boolean">
  <default>true</default>
</key>
<key id="k1" for="edge"
     attr.name="frequency" attr.type="int">
  <desc>frequency of contact in times per week</desc>
  <default>1</default>
</key>
...
<graph edgedefault="directed">
  ...
  <edge id="e7" source="v0" target="v1"/>
  <edge id="e11" source="v0" target="v2">
   <data key="k0">false</data>
    <data key="k1">7</data>
  </edge>
...
```

Fig. 6. GraphML fragment representing two edges, one confirmed with a unit value and the other unconfirmed with a value of seven. The first edge label is a standard attribute stored by visone, the other is user-defined.

Besides GraphML, import and export in a number of simple formats and some formats customary in social network analysis and graph drawing are supported. To communicate results, visualizations can be exported in Scalable Vector Graphics (SVG) or PostScript format. Many conversion tools exist for both. The SVG export routine has been adopted into the core LEDA package.

There is neither a macro language nor an interface for third-party extensions, but limited support of command-line options for batch-mode operations is planned in the future.

5 Examples

We illustrate the intended usage of visone by three exemplary studies in which predecessors of the system have been used to explore and analyze network data.

Drug policy. This project [25] studies the presence of HIV-preventive measures for IV-drug users in nine selected German municipalities. The substantive question underlying this research is, why municipalities with comparable problem pressure differ significantly in the provision of HIV-preventive measures such as methadone substitution or needle exchange.

The policy networks under scrutiny comprise all local organizations directly or indirectly involved in the provision of such measures. In each of the nine municipalities, the 22–38 actors included in the study were queried about relations such as strategic collaboration, common activities, or informal communication with other organizations in the same municipality. None of the networks has more than 120 edges of the same type, and typically more than 50% of them are unconfirmed.

Figure 7 is a typical example of such a network visualized with visone. Note that centrality indices provide insight into the social and political structure of policy making and help understanding the policy outcomes produced.

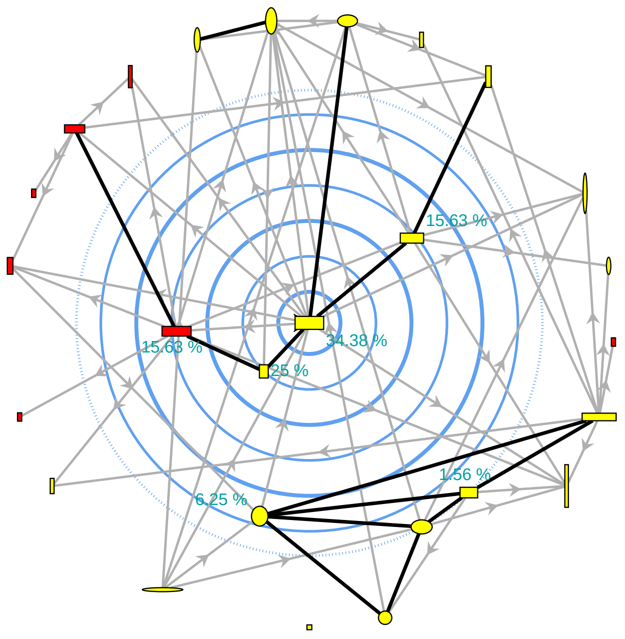


Fig. 7. Organizations involved in drug policy making. Radial visualization of betweenness centrality in network of informal communication. Organizations either have a supportive (yellow) or a repressive (red) attitude towards drug users, and they are public (rectangles) or private (ellipses). Height, width, and area indicate indegree, outdegree, and degree when unconfirmed edges are counted as directed along the claim of existence.

Industry privatization. The second study [31] deals with networks of public, societal and private organizations that developed during the privatization of industrial conglomerates in East Germany as part of the economic transformation after German unification in 1990. Their privatization is understood as political bargaining processes between actors that are connected by ties such as exchange of resources, command, or consideration of interest.

The privatization was foreseen to be carried out by the Treuhandanstalt, a public agency of the federal government. Due to its institutional position and its ownership of all companies, it was generally assumed to be one of the most powerful actors in the transformation of East Germany.

As part of the analysis, status indices are used as indicators for the power or influence of actors. Since the specific index considered for these networks [16, p. 35ff] is not yet provided in visone, it was imported from another software tool (STRUCTURE [17]). Figure 8 gives a visualization example showing whose interests actors involved in the privatization of the ship-building industry claim that they considered.

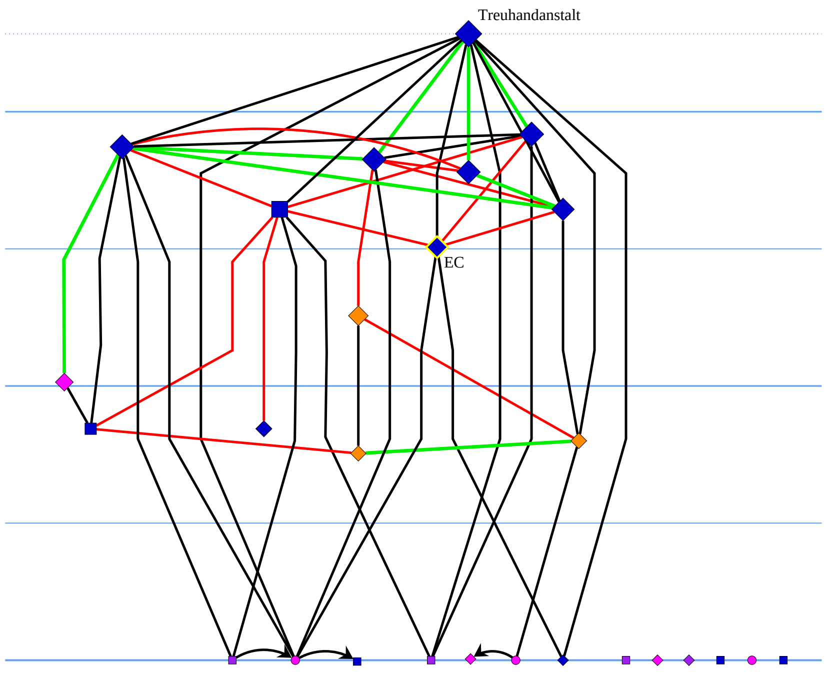


Fig. 8. Interest consideration among actors involved in privatization of the East German ship-building industry after German unification (redrawn from [14], for clarity black and red code edge directions up and down, while green edges are bidirected). Vertex color and shape code additional attributes.

Topic identification. Our third example illustrates the use of methods from social network analysis in another domain, namely topic identification in texts by centering resonance analysis [18]. The structure of texts is represented by graphs that have a vertex for each word occurring in a noun phrase and an edge for each pair of words that appear together in the same noun phrase or consecutively in the same sentence. It is argued that words corresponding to nodes with high betweenness centrality in such a graph are important for the structure of the text and thus a proxy for its topic.

This method was applied to Reuters news dealing with the terrorist attacks of September 11, 2001 [5] to identify, among other things, the main topics, topic changes, side stories, etc. in the news. Figure 9 shows the main topics identified for the very first day of media coverage.

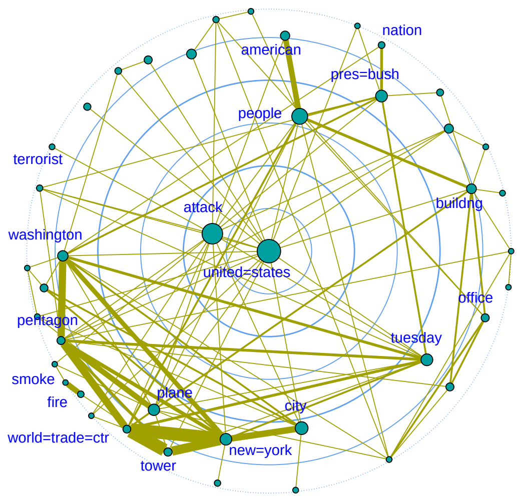


Fig. 9. Text structure in Reuters press releases following the 9/11 terrorist attacks. The news body of more than 46,000 words on the first day leads to a graph with more than 2,400 vertices, of which the 42 most central are shown. Thickness of lines indicates the number of co-occurrences (minimum of two).

6 Software

The visone software is provided as a standalone executable for systems running Linux, Solaris, or Windows, and is free for academic purposes.

Technically, the user interface inherits from LEDA's graph editor class `GraphWin`, though several internal modifications were necessary to address the needs of researchers and students in the social sciences. With application-specific terminology it comprises the usual drawing canvas with pull-down and context (pop-up) menus. Therefore, network data can be imported from a file but also input or edited graphically.

Aside from the analytic and layout procedures described above, we provide some non-standard user interaction facilities as shown in Figure 10. Most importantly, extensive attribute-based selection mechanisms facilitate exploration of data by hiding, adding, or altering objects. For example, users can select vertices based on labels, attributes, graphical attributes, or the selection status of incident edges. These criteria can be combined with an existing selection in various ways. Moreover, the attributes interpreted as strengths and lengths in our unified network model can be switched and modified interactively.

Analytic routines are always applied to the data currently seen by the user, except for unconfirmed edges which are shown for context, but disregarded in the analysis unless they are selected. More precisely, if $E_S \subseteq E$ is the set of selected edges, the social network instance analyzed has edges $E_C \cup E_S$ with lengths or strengths according to the currently displayed edge label. The decision which edges are considered to constitute actual ties and which edges are labeled is thus left to the researcher and can be made on a per-edge basis. Undirected edges displayed in the user interface are internally treated like two oppositely directed edges.

Analysis and layered or radial layout apply to different coordinates of vertex locations, so that it is possible to manually fine-tune the representation of a vertex index or to compare different indices using similar layouts.

Since this is an ongoing project, essentially all components are in development and therefore subject to change. While the layout algorithms currently implemented offer substantial room for further improvement, our long-term

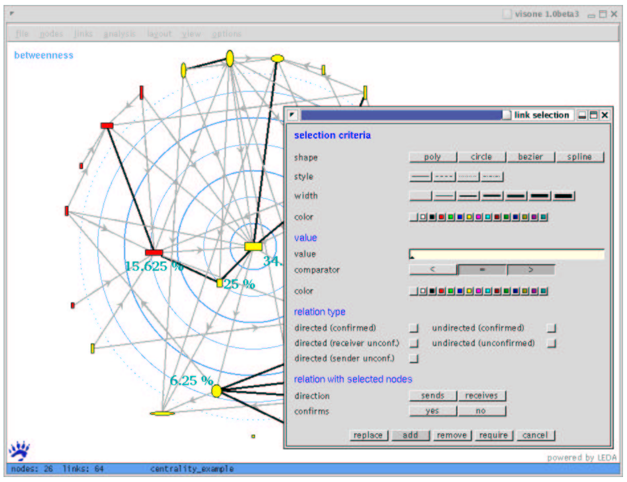


Fig. 10. visone user interface with convenient selection options.

goal is to extend visone with additional visualization modes facilitating, e.g., the comparison of different vertex or edge indices.

References

1. Analytic Technologies. *UCINET V.* Network analysis software. See `http://www.analytictech.com/`
2. J. M. Anthonisse. The rush in a directed graph. Technical Report BN 9/71, Stichting Mathematisch Centrum, Amsterdam, October 1971.
3. A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.
4. V. Batagelj and U. Brandes. Efficient generation of random graphs. Working Paper, 2003.
5. V. Batagelj, U. Brandes, J. C. Johnson, S. Kobourov, L. Krempel, A. Mrvar, and D. Wagner. Analysis and visualization of network data. Special Session during *Sunbelt Social Network Conference XXII*, New Orleans, February 2002.
6. M. Baur and U. Brandes. An improved heuristic for crossing minimization in circular layouts. Working Paper, 2003.
7. M. A. Beauchamp. An improved index of centrality. *Behavioral Science*, 10:161–163, 1965.
8. P. Bonacich. Factoring and weighting approaches to status scores and clique identification. *Journal of Mathematical Sociology*, 2:113–120, 1972.
9. U. Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25(2):163–177, 2001.
10. U. Brandes, M. Eiglsperger, I. Herman, M. Himsolt, and M. S. Marshall. GraphML progress report: Structural layer proposal. In: P. Mutzel, M. Jünger, and S. Leipert (eds.) *Proceedings 9th International Symposium on Graph Drawing (GD '01)*, Springer Lecture Notes in Computer Science 2265:501–512, 2002. For up-to-date information see `http://graphml.graphdrawing.org/`.
11. 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.
12. U. Brandes, P. Kenis, and D. Wagner. Communicating centrality in policy network drawings. *IEEE Transactions on Visualization and Computer Graphics*, 9(2), 2003. To appear.
13. U. Brandes and B. Köpf. Fast and simple horizontal coordinate assignment. In: P. Mutzel, M. Jünger, and S. Leipert (eds.) *Proceedings 9th International Symposium on Graph Drawing (GD '01)*, Springer Lecture Notes in Computer Science 2265:31–44, 2002.
14. U. Brandes, J. Raab, and D. Wagner. Exploratory network visualization: Simultaneous display of actor status and connections. *Journal of Social Structure*, 2(4), 2001.
15. 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.
16. R. S. Burt. *Toward a Structural Theory of Action: Network Models of Social Structure, Perception, and Action.* Academic Press, 1982.
17. R. S. Burt. *Structure, Version 4.2.* Center for the Social Sciences, Columbia University, New York, 1991. See `http://gsbwww.uchicago.edu/fac/ronald.burt/teaching/`

18. S. R. Corman, T. Kuhn, R. D. McPhee, and K. J. Dooley. Studying complex discursive systems: Centering resonance analysis of communication. *Human Communication Research*, 28(2):157–206, 2002.
19. L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40:35–41, 1977.
20. E. N. Gilbert. Random graphs. *The Annals of Mathematical Statistics*, 30(4):1141–1144, 1959.
21. P. Hage and F. Harary. Eccentricity and centrality in networks. *Social Networks*, 17:57–63, 1995.
22. K. M. Hall. An r-dimensional quadratic placement algorithm. *Management Science*, 17(3):219–229, 1970.
23. D. Harel and Y. Koren. Graph drawing by high-dimensional embedding. In: M. T. Goodrich and S. G. Kobourov (ed.) *Proceedings 10th International Symposium on Graph Drawing (GD '02)*, Springer Lecture Notes in Computer Science 2528:207–219, 2002.
24. L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18:39–43, 1953.
25. P. Kenis. An analysis of cooperation structures in local drug policy in germany, 1998. Unpublished Report.
26. J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the Association for Computing Machinery*, 46(5):604–632, 1999.
27. Y. Koren, L. Carmel, and D. Harel. ACE: A fast multiscale eigenvectors computation for drawing huge graphs. In: *Proceedings IEEE Symposium on Information Visualization (InfoVis '02)*, pages 137–144, 2002.
28. S. Masuda, T. Kashiwabara, K. Nakajima, and T. Fujisawa. On the $\mathcal{NP}$-completeness of a computer network layout problem. *Proceedings IEEE International Symposium on Circuits and Systems*, pages 292–295, 1987.
29. C. Matuszewski, R. Schönfeld, and P. Molitor. Using sifting for k-layer straightline crossing minimization. In: J. Katochvíl (ed.) *Proceedings 7th International Symposium on Graph Drawing (GD '99)*, Springer Lecture Notes in Computer Science 1731:217–224, 1999.
30. K. Mehlhorn and S. Näher. *The LEDA Platform of Combinatorial and Geometric Computing*. Cambridge University Press, 1999.
31. J. Raab. *Steuerung von Privatisierung*. Westdeutscher Verlag, 2002.
32. G. Sabidussi. The centrality index of a graph. *Psychometrika*, 31:581–603, 1966.
33. J. M. Six and I. G. Tollis. Circular drawings of biconnected graphs. In: M. T. Goodrich and C. C. McGeoch (ed.), *Proceedings 1st Workshop on Algorithm Engineering and Experimentation (ALENEX '99)*, Springer Lecture Notes in Computer Science 1619:57–73, 1999.
34. J. M. Six and I. G. Tollis. A framework for circular drawings of networks. In: J. Katochvíl (ed.) *Proceedings 7th International Symposium on Graph Drawing (GD '99)*, Springer Lecture Notes in Computer Science 1731:107–116, 1999.
35. T. W. Valente and R. K. Foreman. Integration and radiality: Measuring the extent of an individual's connectedness and reachability in a network. *Social Networks*, 20(1):89–105, 1998.
36. D. J. Watts and S. H. Strogatz. Collective dynamics of "small-world" networks. *Nature*, 393:440–442, 1998.