Erschienen in: Algorithmics of Large and Complex Networks / Lerner, Jürgen; Wagner, Dorothea; Zweig, Katharina A. (Hrsg.). - Berlin, Heidelberg : Springer Berlin Heidelberg, 2009. - (Lecture Notes in Computer Science ; 5515). - S. 330-358. - ISBN 978-3-642-02093-3
https://dx.doi.org/10.1007/978-3-642-02094-0_16

Group-Level Analysis and Visualization of Social Networks

Michael Baur[1], Ulrik Brandes[2], Jürgen Lerner[2], and Dorothea Wagner[1]

[1] Faculty of Informatics, Universität Karlsruhe (TH), KIT
[2] Department of Computer & Information Science, University of Konstanz

Abstract. Social network analysis investigates the structure of relations amongst social actors. A general approach to detect patterns of interaction and to filter out irregularities is to classify actors into groups and to analyze the relational structure between and within the various classes. The first part of this paper presents methods to define and compute structural network positions, i. e., classes of actors dependent on the network structure. In the second part we present techniques to visualize a network together with a given assignment of actors into groups, where specific emphasis is given to the simultaneous visualization of micro and macro structure.

1 Network Analysis

Social network analysis (SNA) [54] is an established, active, and popular research area with applications in sociology, anthropology, organizational studies, and political science, to name a few. In a nutshell, SNA analyzes the structure of relations among (social, political, organizational) actors. While the type and interpretation of actors and relations—as well as the theoretical background of network analysis—varies from application to application, many network analysis *methods* are nevertheless applicable in rather general settings.

In order to abstract from the particular application context, we assume that networks are represented by *graphs* $G = (V, E)$, where V is a set of *vertices*, encoding the actors, and E is a set of *edges* (also called *ties* or *links*), encoding the relation among actors. Edges may be directed, undirected, or of mixed type. Furthermore, vertices and edges may have various attributes encoding, e. g., the type of actors or relations as well as the strength of relations.

Network analysis methods can be classified with respect to the level of granularity of the analyzed objects (compare [11]):

- **Element-level** methods analyze properties of individual vertices and edges, such as importance (*centrality*).
- **Group-level** analysis determines specific subsets of vertices. These methods include the computation of densely connected groups (*clustering*) and the computation of structural roles and positions (*blockmodeling* or *role assignment*, see Sect. 2).

– **Network-level** analysis is interested in global properties of the network, such as density, degree-distributions, transitivity, or reciprocity; as well as in the development of random graph models that are plausible for empirical networks.

In this chapter we focus on group-level network analysis. For surveys encompassing all levels of network analysis, see, for instance, [54] and [11]. In the remainder of this section, we briefly introduce a software to analyze and visualize social networks and state common notation. Thereafter, in Sect. 2, we give an overview of state-of-the-art methods for role assignment and present our own contribution to this field. Section 3 details a visualization technique for networks on which a partition of the vertices is already given (e. g., from clustering, role assignments, or extrinsic vertex-attributes) and where the analyst wants to see the interplay between fine-grained (vertex-level) and coarse-grained (group-level) structures.

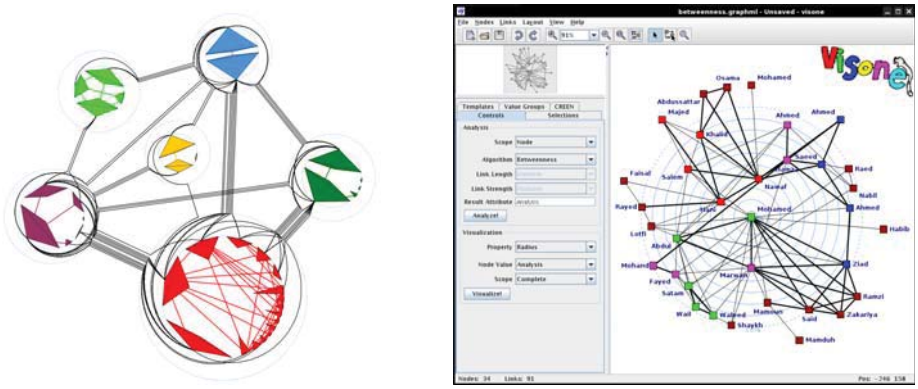
1.1 VISONE – Software for the Analysis and Visualization of Social Networks

Along with the increased relevance of network analysis and the growing size of considered networks, adequate software for social network analysis is becoming more and more important. As part of our project we provide the software tool VISONE[1], aiming to bring together efficient algorithms for methods of analysis and suitable graph drawing techniques for the visualization of networks. Besides our original work, we have included novel algorithms developed by other members of our groups at the universities of Karlsruhe and Konstanz in order to cover fields like centrality indices [10], clusterings [27], and spectral layouts [24]. The functionality is completed by well-known commonly-used methods.

VISONE is not only intended as a testbed for the work of our groups but also as an everyday tool for students and researchers in network analysis. Therefore, we adapt all algorithms to a consistent and comprehensive graph model and put in great efforts to provide a simple but flexible user interface hiding unnecessary complexity. In contrast to common tools which present to the user only a matrix representation of the data, we build on the expressive and explanatory power of graph layouts and provide a complete graphical view of the network (see Fig. 1(b)). Observations indicate that users enjoy the playful nature of our approach.

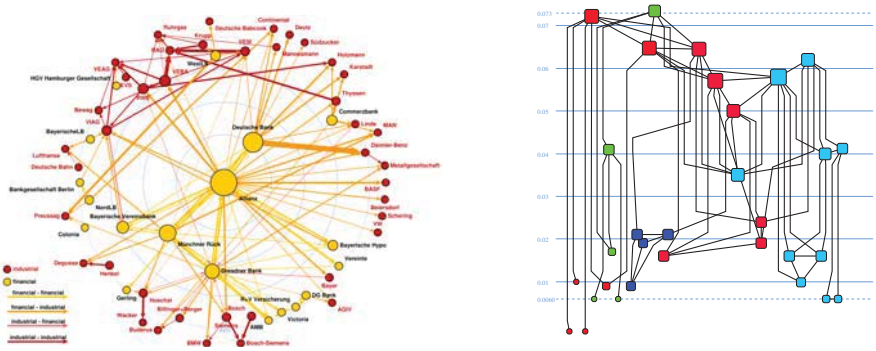
Visualizing social networks is more than simply creating intriguing pictures, it is about generating learning situations: *"Images of social networks have provided investigators with new insights about network structure and have helped them communicate those insights to others"* [25]. Additionally, inappropriate drawings of networks are misleading or at least confusing. Therefore, we pay special attention to the visualization of the networks. Selected general graph layout algorithms provide an uncluttered view on the network and reveal its overall

[1] VISONE is available free of charge for academic and non-commercial purpose from the homepage `http://visone.info`



(a) multi-circular visualization (b) main window of VisoneE

Fig. 1. (a) Multi-circular visualization of a network consisting of six groups. The group structure is clearly visible. Additionally, the height and width of the vertices reflects the number of connections within and between groups. (b) The most notable features of the main window of VisoneE are the large and detailed view of the graph, the small overview, and the control pane on the left hand.



(a) radial visualization (b) status visualization

Fig. 2. Examples of the radial and the status visualization. The positions of the vertices depict centrality measures. Additional information is reflected by the color, shape, size, and width of the vertices and edges.

structure, but the unique feature of VisoneE are the analytic visualizations which exactly depict analysis results, like centrality scores and clusterings, by means of tailored and suggestive graph layouts (see Figs. 1(a) and 2). Combinatorial models of these visualizations allow for the optimization of esthetic properties to improve the expressiveness and exploratory power without changing their analytic signification.

1.2 Basic Notation

Let $G = (V, E)$ be a directed or undirected graph with $n = |V|$ vertices and $m = |E|$ edges. A *partition* of G is a subdivision of the vertex set V into pairwise disjoint, non-empty subsets $V = V_1 \dot\cup \ldots \dot\cup V_k$. In addition to this explicit definition, a partition can be given by an equivalence relation on V or by a surjective mapping $\rho\colon V \to \{1, \ldots, k\}$ of vertices to vertex-classes (called *partition assignment*). These three definitions are mutually in a canonical one-to-one correspondence up to permutation (re-labeling) of classes, see [39], and we typically identify the class i with the set V_i.

A partition assignment $\rho\colon V \to \{1, \ldots, k\}$ defines a smaller graph $Q(G, \rho) = (\mathcal{V}, \mathcal{E})$, called the *quotient graph*, encoding which classes are connected, by setting

$$\mathcal{V} = \{1, \ldots, k\} \text{ and } \mathcal{E} = \{(\rho(u), \rho(v))\,;\ (u, v) \in E\}\ . \tag{1}$$

2 Structural Positions in Networks

The notion of *(structural) position* is fundamental in social network analysis, see for example [54,9]. Actors are said to occupy the same position if they have identical patterns of ties to other actors and the task of determining such classes of actors is referred to as *blockmodeling* or *role assignment*. For instance, by this definition university professors would occupy the same structural position if they have identical patterns of ties to students, secretaries, other professors and so on. Note that this definition of position dependent on the network structure contrasts to more traditional notions of social positions, such as defining the position of professors dependent on the type of contract that they have with their university. In this paper the term *position* always refers to structural position. Various types of role assignment differ in how they operationalize the notion of *identical patterns of ties to other actors* and how they account for deviation from perfectly identical patterns. We continue by reviewing established previous notions for role assignment and outline where the newly proposed *structural similarities* fit in.

2.1 Previous Work on Role Assignments

Basic Notation. A *role assignment* $r\colon V \to \{1, \ldots, k\}$ of a (directed or undirected) graph $G = (V, E)$ is given by a partition of its vertex set V. In context of role assignments, vertex-classes are also referred to as *positions* and the quotient graph is called *role graph*. A role assignment r defines k^2 submatrices, called *blocks*, of G's adjacency matrix A. The block associated to class C and D, denoted by $A[C, D]$, is the $|C| \times |D|$ submatrix of A whose rows correspond to the vertices in C and whose columns correspond to the vertices in D.

If a role graph (i. e., a hypothesis for the role structure of a network) is given, the problem of determining a role assignment that yields this role graph is called a *role assignment problem*, or *prespecified blockmodeling*.

Discrete Approaches. Specific types of role assignments are obtained by requiring that vertex partitions must satisfy specific compatibility constraints with respect to the graph structure. An important distinction between various constraints is whether they require equivalent vertices to be connected to the *same* others —illustrated in Fig. 3 (*left*)—or just to *equivalent* (but not necessarily the same) others—illustrated in Fig. 3 (*right*).

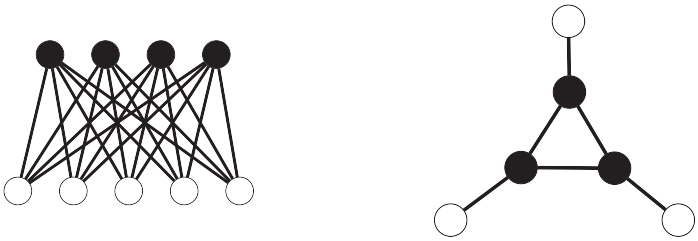


Fig. 3. Two graphs with vertex partitions indicated by the coloring. *Left*: Equivalent vertices have identical neighborhoods. *Right*: Equivalent vertices have equivalent but non-identical neighborhoods.

Neighborhood Identity. The most basic approach defines vertices as *structurally equivalent* [41] if they have identical neighborhoods, i. e., if they are connected to exactly the same others; compare Fig. 3 (*left*). An equivalence is structural if and only if all induced blocks are either complete (consisting only of ones) or zero.

Structural equivalence (SE), however, is too strict and does not well match intuitive notions of network position. Coming back to the example from the beginning of Sect. 2, SE would assign the same position to professors only if they are connected to the *same* students, secretaries, and other professors. Work discussing the insufficiency of SE includes [48], [9], and [42].

Neighborhood Equivalence. To capture more general situations, SE has been relaxed by requiring that vertices occupying the same position must only be connected to the same positions—independent on whether these positions are occupied by the same vertices or different vertices. Thus, two professors would be assigned to the same position if they both have the same patterns of relations to *some* (but not necessarily the same) students, secretaries, and other professors; compare Fig. 3 (*right*) where the black vertices are all connected to some (but different) white vertices. Mathematical formalizations of this idea include *regular equivalence*, *automorphic equivalence*, and *exact regular equivalence*.

A partition is called *regular* [55,22] if for every two of its classes C and D it holds that, whenever one vertex in C has a neighbor in D, then every vertex in C has a neighbor in D. Equivalently, a partition is regular if every induced block is either empty or it has at least one non-zero entry in every row and in every column. A partition is called *exact regular* [22] or *equitable* [31] if for every two of its classes C and D, all vertices in C have the same number of neighbors

in D. Equivalently, a partition is exact regular if for every block B, there are two numbers r_B and c_B such that all rows of B sum up to r_B and all columns of B sum up to c_B. Two vertices u and v are called *automorphically* equivalent if there is a graph automorphism mapping u to v. The notion of automorphic equivalence is quite established in algebraic graph theory (e. g., [31]); work using this concept in social network analysis includes [9]. A structural partition is automorphic, an automorphic partition is equitable, and an equitable partition is regular.

Applicability for Social Network Analysis. The requirement for equitable partitions (and thus for automorphic and structural equivalence) is too strong for social networks (and other irregular, empirical data); due to deviations from ideal structural models, the resulting partitions will have singletons or very small classes. On the other hand, the maximal regular equivalence is often trivial as well; on undirected graphs it corresponds to the division into isolates and non-isolates. Determining non-trivial regular equivalences with a prespecified number of equivalence classes is NP-hard [23]. Regular, equitable, and automorphic equivalence is not robust against the addition or deletion of single edges (e. g., caused by noise or measurement errors); destroying the equivalence of one pair of vertices by adding/deleting an edge can have a cascading effect destroying equivalence of some of their neighbors, second-order neighbors, and so on. In conclusion, structural, automorphic, equitable, and regular partitions have limited applicability for the analysis of empirical data.

Real-valued Degrees of Similarity. To overcome (some of) the abovementioned problems, a formalization of role assignment should not only define *ideal* types of equivalence (such as regular, automorphic, or structural) but also clarify how to measure deviation from ideality (cf. [54]). Seen from a different angle, a formalization of role assignment should not only provide the decision between equivalent and non-equivalent but rather it should yield a *degree of similarity* of vertices.

Relaxing Structural Equivalence to Neighborhood Overlap. Defining degrees of structural equivalence is straightforward, although various different possibilities to do so exist. In most cases similarity is defined by measuring the overlap of the neighborhoods of two vertices and normalizing this measure in an appropriate way. Examples include taking the number of vertices in the intersection of neighborhoods divided by the number of vertices in the union of neighborhoods, the cosine of the angle between the two neighborhood vectors, and the correlation between two neighborhood vectors; see [54] for a more detailed discussion.

The so-defined measures of vertex similarity yield stable and efficient methods for the analysis of social networks. However, they do not overcome the inherent insufficiency of structural equivalence discussed earlier and mentioned in [48], [9], and [42]. In our running example, two professors would only be recognized as similar if they are both in relation to many common others (students, secretaries, and other professors); in contrast, two professors that mostly interact

with disjoint alters would not be assigned similar network positions—even if their patterns of relations are similar.

Relaxing Neighborhood Equivalence. To combine the generality of notions of neighborhood equivalence (e. g., regular, equitable or automorphic partitions) with the robustness and empirical applicability of similarity measures (as opposed to equivalence), there is a need for relaxing neighborhood equivalence. However, previously proposals to do so are unsatisfactory for different reasons. In the following we briefly sketch one proposal for relaxing regular equivalence before we turn to an extended discussion of the newly proposed structural similarities.

Batagelj *et al.* [7] proposed an optimization algorithm to determine, for a fixed number k, a k-partition that has the least number of deviations from regularity; their method belongs to the framework of *generalized blockmodeling* [19]. Recall that a partition is regular if and only if every induced block is either zero or has at least one non-zero entry in each row and in each column. Measuring deviation from regularity of a specific partition is done by counting for each induced block the number of ones on one hand and the number of all-zero rows plus the number of all-zero columns on the other hand. The smaller of these two numbers is considered as the deviation from regularity of this particular block and by summing over all blocks one obtains the deviation from regularity of the partition. The associated optimization problem consists in finding the k-partition with the least deviation from regularity for a given graph. Since it is NP-complete to decide whether a graph admits a regular equivalence relation with exactly k classes [23], the abovementioned optimization problem is NP-hard as well; [7] proposed a local optimization algorithm to compute heuristically a k-partition with a small error. However, this approach is unsatisfactory for its computational inefficiency and lack of understanding of when the algorithm converges to a global optimum. In Sect. 2.2 we propose an alternative relaxation of neighborhood equivalence that enjoys more desirable properties.

2.2 Structural Similarity

The blockmodeling approach from [7] relaxed the *constraint* on partitions from being regular to having the least deviations from regularity. Structural similarities, in contrast, are obtained from equitable partitions (exact regular equivalence) by relaxing the *partitions* and keeping the constraint.

Basic Definitions. A discrete partition of n vertices in k can be represented by its characteristic matrix $P \in \mathbb{R}^{k\times n}$, where the entry $P_{iv} = 1$ if vertex v is in class i and zero else. Thus the degree of membership of a specific vertex to a specific class is either zero or one. Relaxations of partitions are obtained by allowing real-valued degrees of membership.

Definition 1 ([13]). *Given a graph on n vertices, a matrix $P \in \mathbb{R}^{k\times n}$ is called a* projection *(of dimension k) if $PP^{\mathrm{T}} = \mathrm{id}_k$. The entry $P_{iv} = 1$ is called the* degree of membership *of vertex v in class i; a row of P is considered as a* real-valued

class *of vertices in which all n vertices have varying degrees of membership. The real* $n \times n$ *matrix* $S = P^{\mathrm{T}}P$ *is called the associated* similarity*. The entry* S_{uv} *is called the* similarity *of vertices* u *and* v*.*

The uv'th entry of $S = P^{\mathrm{T}}P$ is the inner-product of the two k-dimensional membership vectors of u and v, respectively. This value is large if u and v are to a high degree in the same classes. The constraint $PP^{\mathrm{T}} = \mathrm{id}_k$ on projections ensures that classes are orthogonal (independent) and normalized. Projections and similarities are in a canonical one-to-one correspondence, up to orthogonal transformations of the rows of the projection [13].

Just as a vertex partition defines a smaller graph encoding the adjacency of vertex classes—compare Eq. (1)—a similarity on a graph induces a *quotient* encoding the (weighted) adjacency of (real-valued) classes.

Definition 2 ([13]). *Let* G *be a graph with adjacency matrix* $A \in \mathbb{R}^{n\times n}$ *and* $P \in \mathbb{R}^{k\times n}$ *a projection. Then,* G *and* P *induce a* $k \times k$ *matrix* B *by setting* $B = PAP^{\mathrm{T}}$*. The (weighted) graph* G/P *on* k *vertices that is determined by its adjacency matrix* B *is called the* quotient *of* G *modulo* P*.*

Just as equitable partitions are partitions satisfying a certain compatibility constraint with the network structure, structural similarities are similarities satisfying a structural constraint.

Definition 3 ([13]). *A similarity* S *and its associated projection are called* structural *for a given graph with adjacency matrix* A *if* $SA = AS$*.*

The compatibility constraint $SA = AS$ can be used as an alternative definition of equitable partitions, see [40]. Indeed, if a similarity S is induced by a discrete partition $\mathcal{P}$, then S is structural if and only if $\mathcal{P}$ is equitable [13]. Thus structural similarities do neither relax nor modify the constraint of equitable partitions; they rather generalize discrete partitions to the larger class of similarities.

Characterization and Computation. The key to derive several desirable properties of structural similarities is the following characterization theorem that links structural similarities of a graph G to spectral properties of G's adjacency matrix. General references introducing the use of linear algebra methods in graph theory include [18,31].

Theorem 1 ([13]). *A similarity* S *is structural for an undirected graph with adjacency matrix* A *if and only if the image (i. e., the column-space) of* S *is spanned by eigenvectors of* A*.*

For directed graphs one has to distinguish between similarities that are structural with respect to outgoing edges, incoming edges, or both. Theorem 1 then holds if "spanned by eigenvectors" is replaced by "invariant subspace" and, depending on the type of structurality, "column-space" by "row-space" or "column-space and row-space;" see [40] for details.

Theorem 1 reduces the problem of computing structural similarities to that of computing eigenvectors (or invariant subspaces in the directed case). Many efficient numerical algorithms exist for these problems [32].

2.3 Structural Similarities Compared to Traditional Spectral Techniques

Orthogonal projections to low-dimensional subspaces that are spanned by eigenvectors are a frequent tool in many data analysis and graph partitioning applications. Concrete examples include latent semantic indexing [46], Web search [1], collaborative filtering [4], learning mixtures of distributions [52], analysis of the autonomous systems graph [30], graph clustering [35], random graph coloring [2], spectral graph partitioning [45], and graph bisection [16].

Typically, these methods project onto the eigenvectors corresponding to the (few) eigenvalues with the largest absolute values. (We will refer to these methods as *traditional spectral methods* in the following.) Thus, by Theorem 1, these methods compute special cases of structural similarities; the latter are not restricted to projecting to the largest eigenvalues but can choose all subsets.

We argue below that the difference between these two approaches is conceptually the same as between the requirements of identical vs. equivalent neighborhoods for equivalent vertices (compare Sect. 2.1). Thus, traditional spectral methods can be seen as relaxations of neighborhood *identity*, whereas structural similarities have been characterized as relaxations of equitable partition (exact regular equivalence) and, hence, of neighborhood *equivalence*.

An Illustrating Example. For instance, the (structural) partition shown in Fig. 3 (*left*) can be computed by projecting to the two eigenvalues ± 4.47, which have the maximal absolute values (the others are a seven-fold eigenvalue at 0). In contrast, the (equitable) partition shown in Fig. 3 (*right*) can be computed by projecting to the two eigenvalues 2.41 and -0.41, out of the set of eigenvalues

$$\mathbf{2.41}, 0.62, 0.62, \mathbf{-0.41}, -1.62, -1.62 \ .$$

Thus, restricting spectral algorithms to projections to the maximal eigenvalues yields methods that can not even identify some—intuitively outstanding—automorphic equivalences.

The General Case. Let A be the adjacency matrix of an undirected graph, S the matrix of a structural similarity for this graph, and u and v two vertices. The value $\|A(u-v)\| = \|A(u) - A(v)\|$ is a measure for the difference of the neighborhoods of u and v. Thus, u and v have *almost identical neighborhoods* if $\|A(u-v)\|$ is small. Similarly, u and v are considered as *almost equivalent* by S if $\|S(u-v)\|$ is small. We clarify below that traditional spectral methods optimize the property "$\|S(u-v)\|$ *is small if and only if* $\|A(u-v)\|$ *is small*", i. e.,

> u and v are considered as almost equivalent by S if and only if u and v have almost identical neighborhoods.

To make this precise, let $x_1, \ldots, x_n$ be orthonormalized eigenvectors of A with associated eigenvalues $\lambda_1, \ldots, \lambda_n$ which are ordered such that S projects to the first k eigenvectors. (Thus, if S is determined by traditional spectral methods

the first k eigenvalues are those with maximal absolute values.) Further, let c_1 and c_2 be defined by

$$c_1 = \max_{i=1,\ldots,k} 1/|\lambda_i| \qquad \text{and} \qquad c_2 = \max_{i=k+1,\ldots,n} |\lambda_i| \ .$$

(Note that c_1 is defined only if S does not project to an eigenvalue $\lambda_i = 0$, which can be safely assumed for traditional spectral methods.) If k is given, then traditional spectral methods chose the structural projection of dimension k that minimizes c_1 and c_2 over all structural projections of dimension k. Let y be any vector of norm less than or equal to $\sqrt{2}$ and $y = \sum_{i=1}^{n} a_i x_i$ for uniquely determined real values a_i. It is

$$\|S(y)\|^2 = \sum_{i=1}^{k} a_i^2 \leq c_1^2 \sum_{i=1}^{k} (a_i\lambda_i)^2 \leq c_1^2 \|A(y)\|^2 \qquad \text{and} \tag{2}$$

$$\|A(y)\|^2 = \sum_{i=1}^{k} (a_i\lambda_i)^2 + \sum_{i=k+1}^{n} (a_i\lambda_i)^2 \leq \|A\|_2^2 \|S(y)\|^2 + 2c_2^2 \ . \tag{3}$$

By taking $y = u - v$ for the two vertices u and v, we obtain from (2) and (3) the following two properties for a structural similarity S.

1. Assume that S does not project to an eigenvalue $\lambda_i = 0$. If $\|A(u) - A(v)\|$ is small, then $\|S(u) - S(v)\|$ is small, i. e., vertices with almost identical neighborhoods are considered as almost equivalent by S. Furthermore, the ratio $\|S(u-v)\|/\|A(u-v)\|$ is bounded from above by c_1 which is minimized by traditional spectral methods.
2. Conversely, if $\|S(u)-S(v)\|$ is small then $\|A(u)-A(v)\|$ is bounded by $\sqrt{2}c_2$ plus a small $\varepsilon > 0$, i. e., if vertices are seen as almost equivalent by S, then their neighborhoods can differ by no more than $\sqrt{2}c_2$. Traditional spectral methods minimize c_2, i. e., those methods recognize only vertices with almost identical neighborhoods as almost equivalent.

It is important to note that these two properties can cause traditional spectral methods to miss some structure of the graph. Vertices may have a high structural similarity (e. g., they may be even automorphically equivalent) without having almost identical neighborhoods; compare Fig. 3.

2.4 The Role-Assignment Problem

For a given graph there is a huge set of structural similarities. Selecting the most appropriate one (or at least narrowing the choice) can be done by specifying how vertex classes (corresponding to structural positions in the network) are connected. Section 2.6 illustrates how the following theorem can be applied in the analysis of empirical data.

Theorem 2 ([13]). *Let G be an undirected graph with adjacency matrix $A \in \mathbb{R}^{n\times n}$ and R be a graph with adjacency matrix $B \in \mathbb{R}^{k\times k}$. Then, there is a structural projection $P \in \mathbb{R}^{k\times n}$ such that $B = PAP^{\mathrm{T}}$ if and only if B is symmetric*

and the characteristic polynomial of B divides the characteristic polynomial of A. In this case, the image of the similarity associated to P is generated by eigenvectors of A associated to the eigenvalues of B.

In addition to its practical value, Theorem 2 also shows that the role assignment problem, which is computationally intractable for discrete notions of network position [23], is efficiently solvable for structural similarities.

2.5 Stability and Non-arbitrariness

A structural similarity S is associated to a set of eigenvalues of the graphs adjacency matrix A, namely the eigenvalues of $B = PAP^{\mathrm{T}}$. If all of these eigenvalues have the same multiplicity in B as they have in A, we call S a *simple* structural similarity. We show in this section that simple structural similarities enjoy two properties—that of being invariant under automorphisms and that of depending continuously from the adjacency matrix.

Non-arbitrariness. We say that a similarity (and hence more specifically a partition) is *non-arbitrary* if it is only derived from the graph's structure and not from a particular labeling of vertices. This, in turn, is formalized by being invariant under graph automorphisms, where *invariant under an automorphism* φ means that the similarity of every pair of vertices u and v is the same as the similarity of their images $\varphi(u)$ and $\varphi(v)$ (this is made precise in Def. 4). Figure 4 shows a small network together with an automorphism invariant partition (*left*) and a partition that is not automorphism invariant (*right*).

Definition 4. *Let $G = (V, E)$ be a graph. A similarity S is called* automorphism invariant *(for G) if for every two vertices $u, v \in V$ and every graph automorphism $\varphi\colon V \to V$ of G it is $S_{uv} = S_{\varphi(u)\varphi(v)}$.*

Theorem 3 ([40]). *A simple structural similarity is automorphism invariant.*

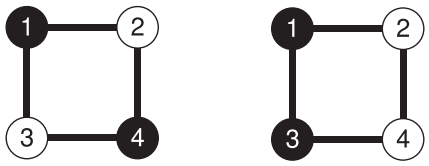


Fig. 4. Two different colorings on a graph with spectrum $\{2, 0, 0, -2\}$. *Left*: The coloring corresponds to the structural projection onto $\{2, -2\}$ and is automorphism invariant. This coloring reflects the unique bipartition of the graph and is therefore well justified by the graph structure. *Right*: The coloring corresponds to a structural projection onto $\{2, 0\}$ (only one eigenvector with eigenvalue 0 is taken). This coloring is not automorphism invariant (e. g., transposing 2 and 3 changes the partition). Intuitively, it seems to be arbitrary and not justifiable by the graph structure that Vertex 1 should be more similar to 3 than to 2, as suggested by the partition on the right.

The converse of Theorem 3 would hold if we took a weaker definition for automorphisms, see [40]. Theorem 3 also gives a criteria when equitable partitions are automorphism invariant since these are special cases of structural similarities.

Stability. A further desirable property of structural similarities is that their robustness to changes in the input data (e. g., caused by errors or dynamics) can be well-characterized. The following definition corresponds to the definition of the *separator*, known in matrix perturbation theory [51].

Definition 5. *Let S be a simple structural similarity for an undirected graph with adjacency matrix A. Let B be the induced quotient, Λ_B the spectrum of B, and Λ_A the spectrum of A. The positive real number*

$$\sigma(S) = \min\{|\lambda_1 - \lambda_2| \,;\, \lambda_1 \in \Lambda_B,\, \lambda_2 \in \Lambda_A \setminus \Lambda_B\}$$

is called the stability *of S.*

For a more general definition including the case of directed graphs see [51]. A large value $\sigma(S)$ guarantees resistance to perturbations of the input matrix A. Many error bounds can be given differing in the matrix norms that are used to measure the deviation and in the assumptions on the form of the error. See [51, Chapts. IV and V] for a representative set of error bounds. Examples of concrete error bounds for structural similarities under different assumptions are given in [12] and [14].

2.6 Applications of Structural Similarity

A structural similarity yields a low-dimensional embedding for the vertices of a graph. There are several ways for post-processing this embedding to obtain insights into the data. The first way is to apply a distance-based clustering procedure to the vertices in the low-dimensional embedding to obtain a discrete vertex partition. We followed this approach in [14], where it has been shown that the framework of structural similarities yields more general algorithms for random graph coloring. While traditional approaches can only deal with random graph models where edge probabilities are uniform, the newly proposed algorithm can handle models with non-uniform probabilities, provided that each vertex has the same expected number of neighbors from each class of differently colored vertices. This generalizations is conceptually the same as the relaxation from neighborhood identity to neighborhood equivalence; compare Sect. 2.1 and Sect. 2.3.

A second way to deal with the low-dimensional embedding is not to round it to a discrete partition but rather to apply multidimensional scaling (MDS) techniques to *visualize* the result in two or three dimensional space. Vertices that occupy (almost) the same positions will then be drawn close together and vertices that occupy very different positions will be far apart. The advantage of such a continuous representation of vertex positions is that we can accommodate with vertices that stand between two or more positions (that play more than one role).

We argue that such situations arise often in real-world data and forcing vertices to be members of one and only one class would then produce sub-optimal results. We will follow this approach to develop analysis and visualization methods for conflict networks.

Analysis and Visualization of Conflict Networks. The framework of structural similarities is especially convenient to develop methods for the analysis of large, noisy, empirical data sets. In [12] we presented a method to visualize dynamic networks of conflict between political actors. We review its essentials here, since this method is a good way to illustrate the use of Theorem 2.

Conflict networks are networks where the edges have a negative or hostile interpretation, such as criticism, accusations, or military engagements. Weighted edges arise from time-stamped events between the actors involved. Given a conflict network we generate a dynamic visualization that shows which group of actors is in opposition to which other group, which actors are most involved in conflict, and how do conflicts emerge, change their structure, and fade out over time. The example data set is from the Kansas Event Data System (KEDS) [49] and consists of approximately 78,000 dyadic events between political actors in the Balkans region.

We make the assumption that actors are loosely grouped together such that conflicts occur mostly between members of different groups. Thus, an actor is a member of one out of k classes to the extent that it has conflicts with members of the other classes.

We describe our method for the situation when there are only two groups that are mutually in conflict. To obtain a real-valued assignment of actors to the two groups we consider the quotient R_{cw} shown in Fig. 5 (*left*). The eigenvalues of R_{cw} are

$$\lambda = c + w \qquad \text{and} \qquad \mu = c - w \ .$$

From a different perspective the edge-weights of the quotient R_{cw} are determined by its two eigenvalues λ and μ as

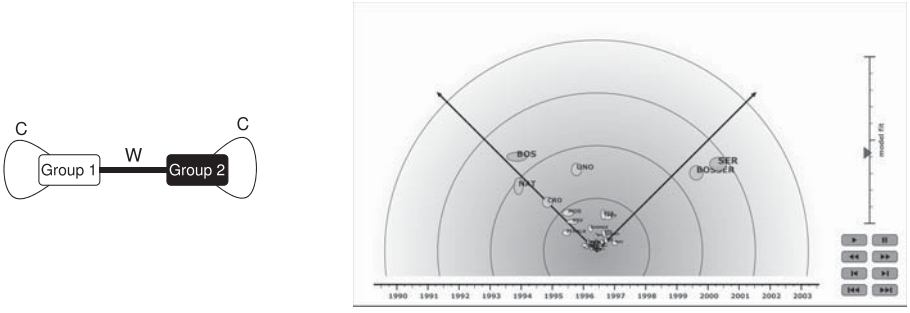


Fig. 5. *Left:* Quotient of a 2-dimensional conflict space. *Right:* Conflictive groups in the Balkans for the period from 1989 until 2003. Actors are mapped into the left(right) dimension to the extent that they are members of one of the two groups. The distance from the origin is a measure of how *involved* actors are in conflict.

$$c = \frac{\lambda + \mu}{2} \qquad \text{and} \qquad w = \frac{\lambda - \mu}{2} \ .$$

Theorem 2 implies that a similarity S is structural with $G/S = R_{cw}$, if and only if S is the projection onto the eigenvalues λ and μ of R_{cw}. Since our goal is to maximize the edge weight between the clusters, i. e., to maximize w, the optimal choice are the largest and the smallest eigenvalue of the adjacency matrix.

To obtain the actual degrees of membership to the two groups, the appropriate basis for the two-dimensional image space has to be identified. In short, the matrix P whose rows are the two eigenvectors has to be rotated by the inverse eigenvector-basis of R_{cw} (details can be found in [12]). An example for a projection to conflict space can be seen in Fig. 5 (*right*). As it can be seen, actors have largely differing degrees of membership and can also stand between groups. It has been shown in [15] how this method can be extended to more than two groups.

To show the development over time, we defined in [12] time-dependent conflict networks that take into account only the events within a certain time-frame. By letting this time-frame move forward, we obtain a smoothly animated visualization showing the development of conflicts over time.

3 Multi-circular Visualization

An important aspect in the visualization of many types of networks is the interplay between fine- and coarse-grained structures. While the micro-level graph is given, a macro-level graph is induced by a partitioning of the micro-level vertices. For example it may originate from a group-level network analysis such as a clustering or may just be given in advance.

We propose a tailored visualization for networks with such a micro/macro structure based on a novel multi-circular drawing convention. Given a layout of the macro-level graph with large nodes and thick edges, each vertex of the micro-level graph is drawn in the area defined by the macro-vertex it belongs to, and each micro-edge is routed through its corresponding macro-edge. In more detail, each micro-vertex is placed on a circle inside of the area of its corresponding macro-vertex and micro-edges whose end vertices belong to the same macro-vertex are drawn inside of these circles. All other micro-edges are then drawn inside of their corresponding macro-edges and at constant but different distances from the border of the macro-edge, i. e., in straight-line macro-edges they are drawn as parallel lines. These edges must also be routed inside the area of macro-vertices to connect to their endpoints, but are not allowed to cross the circles. Figure 6 shows a concrete example of this model. Micro-edges connecting vertices in the same macro-vertex are drawn as straight lines. Inside of macro-vertices, the other edges spiral around the circle of micro-vertices until they reach the area of the macro-edge. We give a combinatorial description of the above model and then focus on the algorithmically most challenging aspect of these layouts, namely crossing reduction by cyclic ordering of micro-vertices and choosing edge winding within macro-vertices.

(a) geometric grouping and straight-line edges (b) multi-circular layout (c) corresponding macro graph

Fig. 6. (a) Example organizational network with geometric grouping and straight-line edges (redrawn from [37]). In our multi-circular layout (b), all details are still present and the macro-structure induced by the grouping becomes clearly visible. Additionally, the height and width of the vertices reflects the number of connections within and between groups.

We do not impose restrictions on the macro-level layout other than sufficient thickness of edges and vertices, so that the micro-level graph can be placed on top of the macro-level graph, and provide layout algorithms and tailored means of interaction to support the generation of appropriate macro-layouts.

While the drawing convention consists of proven components—geometric grouping is used, e. g., in [37,50], and edge routing to indicate coarse-grained structure is proposed in, e. g., [6,34]—our approach is novel in the way micro-vertices are organized to let the macro-structure dominate the visual impression without cluttering the micro-level details too much. Note also that the setting is very different from layout algorithms operating on structure-induced clusterings (e. g., [3,36]), since no assumptions on the structure of clusters are made (they may even consist of isolates). Therefore, we neither want to utilize the clustering for a better layout, nor do we want to display the segregation into dense subregions or small cuts. Our aim is to represent the interplay between a (micro-level) graph and a (most likely extrinsic) grouping of its vertices.

After defining some basic terminology in Sect. 3.1, we state required properties for macro-graph layout in Sect. 3.2 and recapitulate related micro-layout models in Sect. 3.3. Multi-circular micro-graph layout is discussed in more detail in Sect. 3.4 and crossing reduction algorithms for it are given in Sect. 3.5.

3.1 Preliminaries

Throughout this section, we restrict ourselves to simple undirected graphs. In the following, let $E(v) = \{\{u, v\} \in E;\ u \in V\}$ denote the incident edges of a vertex $v \in V$, let $N(v) = \{u \in V;\ \{u, v\} \in E\}$ denote its neighbors, and let $\operatorname{sgn} : \mathbb{R} \to \{-1, 0, 1\}$ be the signum function.

Since each micro-vertex is required to belong to exactly one macro-vertex, the macro-structure defines a partition assignment $\rho : V \to \{1, \ldots, k\}$ and a

prototypical macro-graph is the corresponding quotient graph $Q(G, \rho)$. An edge $\{u, v\} \in E$ is called an *intra-partition edge* if and only if $\rho(u) = \rho(v)$, and *inter-partition edge* otherwise. The set of intra-partition edges of a partition V_i is denoted by E_i, the set of inter-partition edges of two partitions V_i, V_j by $E_{i,j}$. We use $G = (V, E, \rho)$ to denote a graph $G = (V, E)$ and a related partition assignment ρ.

A *circular order* $\pi = \{\pi_1, \ldots, \pi_k\}$ defines for each partition V_i a vertex order π_i as a bijective function $\pi_i : V_i \to \{1, \ldots, |V_i|\}$ with $u \prec v \Leftrightarrow \pi_i(u) < \pi_i(v)$ for any two vertices $u, v \in V_i$. An order π_i can be interpreted as a counter-clockwise sequence of distinct positions on the circumference of a circle.

3.2 Macro Layout

No specific layout strategy for the macro-graph is required as long as its elements are rendered with sufficient thickness to draw the underlying micro-graph on top of them. In order to achieve this, post-processing can be applied to any given layout [29] or methods which consider vertex size (e. g., [33,53]) and edge thickness (e. g., [20]) have to be used.

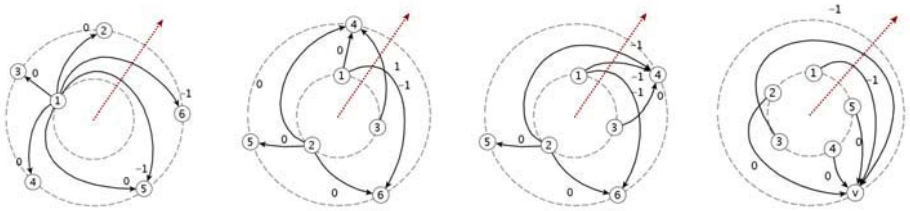
From a macro-layout we get *partition orders* $\Pi_i : N_Q(V_i) \to \{1, .., \deg(V_i)\}$ for each partition V_i, defined by the sequence of its incident edges in $Q(G, \rho)$, and a partition order $\Pi = \{\Pi_1, \ldots, \Pi_k\}$ for G. For each macro-vertex this can be seen as a counter-clockwise sequence of distinct docking positions for its incident macro-edges on its border.

3.3 Related (Micro) Layout

Before we discuss the multi-circular layout model for the micro-graph, let us recall the related concepts of (single) circular and radial embeddings. In *(single) circular layouts* all vertices are placed on a single circle and edges are drawn as straight lines. Therefore, a *(single) circular embedding* ε of a graph $G = (V, E)$ is fully defined by a vertex order π, i. e., $\varepsilon = \pi$ [8]. Two edges $e_1, e_2 \in E$ *cross* in ε if and only if the endvertices of e_1, e_2 are encountered alternately in a cyclic traversal.

In *radial level layouts* the partitions are placed on nested concentric circles (*levels*) and edges are drawn as curves between consecutive partitions. Therefore, only graphs $G = (V, E)$ with a *proper* partition assignment $\rho : V \to \{1, \ldots, k\}$ are allowed, i. e., $|\rho(u) - \rho(v)| = 1$ for all edges $\{u, v\} \in E$. Note that this prohibits intra-partition edges and edges connecting non-consecutive partitions. For technical reasons, edges are considered to be directed from lower to higher levels.

Recently, Bachmaier [5] investigated such layouts. They introduced a *ray* from the center to infinity to mark the start and end of the circular vertex orders. Using this ray, it is also possible to count how often and in which direction an edge is wound around the common center of the circles. We call this the *winding* $\psi : E \to \mathbb{Z}$ of an edge (Bachmaier called this *offset*). $|\psi(e)|$ counts the number of crossings of the edge with the ray and the sign reflects the mathematical direction



(a) some incident edges (b) node 4 is at position 0 (c) node 4 rotated to position 2 (d) without parting

Fig. 7. Examples of Radial layouts. Edges are labeled with their winding value.

of rotation. See Fig. 7 for some illustrations. Finally, a *radial embedding* ε of a graph $G = (V, E, \rho)$ is defined to consist of a vertex order π and an edge winding ψ, i. e., $\varepsilon = (\pi, \psi)$.

There is additional freedom in radial drawings without changing the crossing number: the rotation of a partition V_i. A *rotation* moves a vertex v with extremal position in π_i over the ray. The layout in Fig. 7(c) is a clockwise rotation of the layout in Fig. 7(b). Rotations do not modify the cyclic order, i. e., the neighborhood of each vertex on its radial level is preserved. However, the winding of the edges incident to v and all positions of π_i must be updated.

Crossings between edges in radial embeddings depend on their winding and on the order of the endvertices. There can be more than one crossing between two edges if they have very different windings. The number of crossings between two edges $e_1, e_2 \in E$ in an radial embedding ε is denoted by $\chi_\varepsilon(e_1, e_2)$. The (radial) crossing number of an embedding ε and a level graph $G = (V, E, \rho)$ is then naturally defined as

$$\chi(\varepsilon) = \sum_{\{e_1,e_2\}\in E, e_1\neq e_2} \chi_\varepsilon(e_1, e_2)$$

and $\chi(G) = \min\{\chi(\varepsilon) : \varepsilon \text{ is a radial embedding of } G\}$ is called the *radial crossing number* of G.

Theorem 4 ([5]). *Let $\varepsilon = (\pi, \psi)$ be a radial embedding of a two-level graph $G = (V_1 \dot\cup V_2, E, \rho)$. The number of crossings $\chi_\varepsilon(e_1, e_2)$ between two edges $e_1 = (u_1, v_1) \in E$ and $e_2 = (u_2, v_2) \in E$ is*

$$\chi_\varepsilon(e_1, e_2) = \max\left\{0, \left|\psi(e_2) - \psi(e_1) + \frac{b-a}{2}\right| + \frac{|a|+|b|}{2} - 1\right\},$$

where $a = \operatorname{sgn}(\pi_1(u_2) - \pi_1(u_1))$ and $b = \operatorname{sgn}(\pi_2(v_2) - \pi_2(v_1))$.

Bachmaier also states that in crossing minimal radial embeddings every pair of edges crosses at most once and adjacent edges do not cross at all. As a consequence, only embeddings need to be considered where there is a clear *parting* between all edges incident to the same vertex u. The parting is the position of

the edge list of u that separates the two subsequences with different winding values. See again Fig. 7 for layouts with and without proper parting. Furthermore, only embeddings with small winding are considered because large winding values correspond to very long edges which are difficult to follow and generally result in more crossings.

3.4 Multi-circular Layout

Unless otherwise noted, vertices and edges belong to the micro-level in the following. In the micro-layout model each vertex is placed on a circle inside of its corresponding macro-vertex. Intra-partition edges are drawn within these circles as straight lines. Inter-partition edges are drawn inside their corresponding macro-edges and at constant but different distances from the border of the macro-edge. To connect to their incident vertices, these edges must also be routed inside of macro-vertices. Since they are not allowed to cross the circles, they are drawn as curves around them. Such a drawing is called a *(multi-)circular layout*. Since intra- and inter-partition edges cannot cross, all crossings of intra-partition edges are completely defined by the vertex order π_i of each partition V_i. Intuitively speaking, a vertex order defines a circular layout for the intra-partition edges. In the following we thus concentrate on inter-partition edges.

The layout inside each macro-vertex V_i can be seen as a two-level radial layout. The orders can be derived from the vertex order π_i and the partition order Π_i. Similar to radial layouts a *ray* for each partition is introduced and the beginning of the orders and the edge winding is defined according to these rays. Note that for each edge $e = \{u, v\} \in E$, $u \in V_i$, $v \in V_j$, two winding values are needed, one for the winding around partition V_i denoted by $\psi_i(e) = \psi_u(e)$, and one for the winding around partition V_j denoted by $\psi_j(e) = \psi_v(e)$. If the context implies an implicit direction of the edges, windings are called either source or target windings, respectively. Since radial layouts can be rotated without changing the embedding, rays of different partitions are independent and can be directed arbitrarily. Finally, a *multi-circular embedding* ε is defined by a vertex order π, a partition order Π, and the winding of the edges ψ, i. e., $\varepsilon = (\pi, \Pi, \psi)$.

Observation 5. *For each partition V_i in a multi-circular embedding ε=(π,Π,ψ) a two-level radial embedding $\varepsilon_i = ((\pi_i, \pi'), \psi_i)$ is defined by the vertex order π_i, the partition order Π_i, and the edge winding ψ_i, where $\pi'(v) = \Pi_i(\rho(v)), v \in V \setminus V_i$.*

There is another connection between radial and multi-circular layouts. A two-level radial layout can easily be transformed into a two-partition circular layout and vice versa. Given a graph $G = (V_1 \dot\cup V_2, E, \rho)$ and a radial embedding $\varepsilon = (\pi, \psi)$ of G, the two-partition circular embedding $\varepsilon^* = (\pi^*, \Pi^*, \psi^*)$ defined by $\pi_1^* = \pi_1$, $\pi_2^* = -\pi_2$, $\Pi_1^* = 0$, $\Pi_2^* = 0$, and $\psi_1^*(e) = \psi(e)$, $\psi_2^*(e) = 0$ realizes exactly the same crossings (see Fig. 8 for an example). Intuitively speaking, the topology of the given radial embedding is not changed if the two circles are dragged apart and one of the vertex orders is reversed. If a two-partition circular

embedding $\varepsilon^* = (\pi^*, \Pi^*, \psi^*)$ is given, a related radial embedding $\varepsilon = (\pi, \psi)$ is defined by $\pi_1 = \pi_1^*$, $\pi_2 = -\pi_2^*$, and $\psi(e) = \psi_1(e) - \psi_2(e)$.

Observation 6. *There is a one-to-one correspondence between a two-level radial embedding and a two-circular embedding.*

Crossings in the micro-layout are due to either the circular embedding or crossing macro-edges. Since crossings of the second type cannot be avoided by changing the micro-layout, they are not considered in the micro-layout model. Obviously, pairs of edges which are not incident to a common macro-vertex can only cause crossings of this type. For pairs of edges which are incident to at least one common macro-vertex corresponding two-level radial layouts are defined using Observations 5 and 6 and the number of crossings are computed by modifications of Theorem 4.

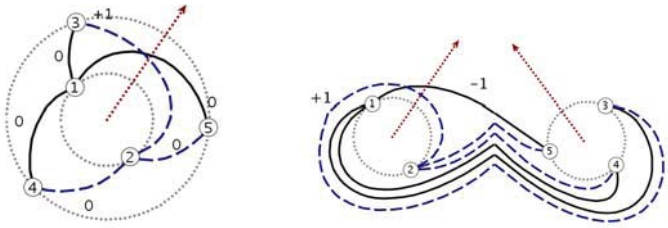


Fig. 8. A two-level radial layout and its corresponding two-circular layout

Theorem 7. *Let $\varepsilon = (\pi, \Pi, \psi)$ be a multi-circular embedding of a graph $G = (V, E, \rho)$ and let $e_1 = \{u_1, v_1\}$, $e_2 = \{u_2, v_2\} \in E$ be two inter-partition edges. If e_1 and e_2 share exactly one common incident macro-vertex, e. g., $V_i = \rho(u_1) = \rho(u_2)$, $\rho(v_1) \neq \rho(v_2)$, then the number of crossings of e_1 and e_2 is*

$$\chi_\varepsilon(e_1, e_2) = \max\left\{0, \left|\psi_i(e_2) - \psi_i(e_1) + \frac{b-a}{2}\right| + \frac{|a|+|b|}{2} - 1\right\},$$

where $a = \operatorname{sgn}(\pi_i(u_2) - \pi_i(u_1))$ and $b = \operatorname{sgn}(\Pi(\rho(v_2)) - \Pi(\rho(v_1)))$.

Proof. Let $e_1 = \{u_1, v_1\}$, $e_2 = \{u_2, v_2\} \in E$ be two edges with exactly one common end partition, e. g., $V_i = \rho(u_1) = \rho(u_2), \rho(v_1) \neq \rho(v_2)$. All crossings between e_1 and e_2 not caused by the macro layout occur in the macro-vertex V_i. According to Observation 5, the fraction of the layout in V_i can be regarded as a two-level radial layout defined by $\varepsilon' = (\pi_i, \Pi_i \circ \rho)$. Applying Theorem 4 to the embedding ε', the theorem follows. □

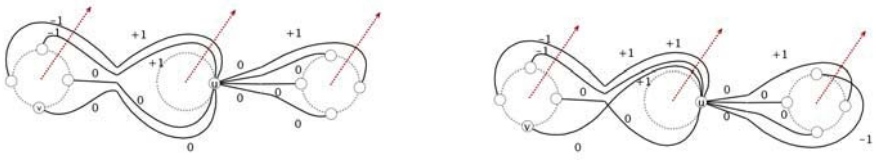
Theorem 8. *Let $\varepsilon = (\pi, \Pi, \psi)$ be a multi-circular embedding of a graph $G = (V, E, \rho)$ and let $e_1 = \{u_1, v_1\}$, $e_2 = \{u_2, v_2\} \in E$ be two inter-partition edges. If e_1 and e_2 belong to the same macro-edge, e. g., $V_i = \rho(u_1) = \rho(u_2)$, $V_j = \rho(v_1) = \rho(v_2)$, then the number of crossings between e_1 and e_2 is*

$$\chi_\varepsilon(e_1, e_2) = \max\left\{0, \left|\psi'(e_2) - \psi'(e_1) + \frac{b-a}{2}\right| + \frac{|a|+|b|}{2} - 1\right\},$$

where $a = \operatorname{sgn}(\pi_i(u_2) - \pi_i(u_1))$, $b = \operatorname{sgn}(\pi_j(v_1) - \pi_j(v_2))$ *and* $\psi'(e) = \psi_i(e) + \psi_j(e)$.

Proof. Let $e_1, e_2 \in E$ be two inter-partition edges which belong to the same macro-edge. Since only two partitions are involved, a two-level radial embedding ε' for e_1 and e_2 can be defined according to Observation 6. In ε' the two edges e_1 and e_2 cause the same crossings than in ε. Applying Theorem 4 to the embedding ε', the theorem follows. □

Similar to radial layouts, in a crossing minimal multi-circular embedding incident edges do not cross and there is at most one crossing between every pair of edges. Therefore, only embeddings need to be considered where there is a clear *parting* between all edges incident to the same vertex $u \in V_i$. Since in multi-circular layouts winding in different macro-vertices can be defined independently, the edge list $E(u)$ of u is split by target partitions resulting in edge lists $E(u)_j = \{\{u, v\} \in E(u) : v \in V_j\}$. For each list $E(u)_j$, a position ℓ_j separates the two subsequences with different values of winding ψ_j and defines the parting for this partition. Furthermore, there is also a parting for V_i defined on the edge list $E(u)$. The order of $E(u)$ for this parting depends on the partings ℓ_j in the target partitions V_j. Edges are sorted by the partition order and for edges to the same partition V_j, ties are broken by the reverse vertex order started not at the ray but at the parting position ℓ_j. Then, the parting for V_i is the position ℓ_i which separates different values of winding ψ_i in the so ordered list. See Fig. 9 for a layout with parting and a layout where the edge $\{u, v\}$ violates the parting.



(a) parting (b) edge $\{u, v\}$ violates parting

Fig. 9. Not all winding combinations for the incident edges of u result in a good layout

Corollary 1. *Multi-circular crossing minimization is* $\mathcal{NP}$*-hard.*

Proof. Single circular and radial crossing minimization [5,43] are $\mathcal{NP}$-hard. As we have already seen, these two crossing minimization problems are subproblems of the multi-circular crossing minimization problem, proving the corollary.

As a consequence, we do not present exact algorithms for crossing minimization in multi-circular layouts. Instead, we propose extensions of some well-known crossing reduction heuristics for horizontal and radial crossing reduction.

3.5 Layout Algorithms

Since the drawing of inter-partition edges inside a macro-vertex can be seen as a radial drawing, a multi-circular layout can be composed of separate radial layouts for each macro-vertex (for instance using the techniques of [5,28,50]). However, such a decomposition approach is inappropriate since intra-partition edges are not considered at all and inter-partition edges are not handled adequately due to the lack of information about the layout at the adjacent macro-vertices. For example, choosing a path with more crossings in one macro-vertex can allow a routing with much less crossings on the other side.

Nevertheless, we initially present in this section adaptations of radial layout techniques because they are quite intuitive, fast, and simple, and can be used for the evaluation of more advanced algorithms.

Barycenter and Median Layouts. The basic idea of both the barycenter and the median layout heuristicis the following: each vertex is placed in a central location computed from the positions of its neighbors - in either the barycenter or the median position - to reduce edge lengths and hence the number of crossings. For a two-level radial layout, the *Cartesian Barycenter* heuristic gets the two levels and a fixed order for one of them. All vertices of the fixed level are set to equidistant positions on a circle and the component-wise barycenter for all vertices of the second level is computed. The cyclic order around the center defines the order of the vertices and the edges are routed along the geometrically shortest-path. The *Cartesian Median* heuristic is defined similar. Running time for both heuristics is in $\mathcal{O}(|E| + |V| \log |V|)$.

Both heuristics are easily extended for multi-circular layouts. The layout in each macro-vertex V_i is regarded as a separate two-level radial layout as described in Observation 6 and the partition orders Π_i are used to define the orders of the fixed levels. Because of the shortest-path routing, no two edges cross more than once and incident edges do not cross at all in the final layout. On the other hand, the used placement and winding strategies are based on edge length reduction and avoid crossings only indirectly.

Multi-circular Sifting. In order to overcome the drawbacks of the radial layout algorithms described before, we propose an extension of the sifting heuristic which computes a complete multi-circular layout and considers edge crossings for optimizing both vertex order and edge winding, and thus is expected to generate better layouts.

Sifting was originally introduced as a heuristic for vertex minimization in ordered binary decision diagrams [47] and later adapted for the layered one-sided, the circular, and the radial crossing minimization problems [5,8,44]. The idea is to keep track of the objective function while moving a vertex along a fixed order of all other vertices. The vertex is then placed in its (locally) optimal position. The method is thus an extension of the greedy-switch heuristic [21]. For crossing reduction the objective function is the number of crossings between the edges incident to the vertex under consideration and all other edges. In multi-circular layouts this function depends on both the vertex order and

the edge winding. Therefore, for each position of a vertex, the winding values for its incident edges which result in the minimal crossing number have to be identified.

The efficient computation of crossing numbers in sifting for layered and single circular layouts is based on the locality of crossing changes, i. e., swapping consecutive vertices $u \curvearrowright v$ only affects crossings between edges incident to u with edges incident to v. In multi-circular layouts this property clearly holds for intra-partition edges since they form (single-)circular layouts. For inter-partition edges the best routing path may require an update of the windings. Such a change can affect crossings with all edges incident to the involved partitions.

Since swapping the positions of two consecutive vertices (and keeping the winding values) only affects incident edges, the resulting change in the number of crossings can be computed efficiently. Therefore, an efficient strategy for updating edge windings while $u \in V_i$ moves along the circle is needed. Instead of probing each possible combination of windings for each position of u the parting of the edge lists is considered. Note that the parting for the source partition and all the partings for the target partitions have to be simultaneously altered because for an edge, a changed winding in the source partition may allow a better routing with changed winding in the target partition. Intuitively speaking, the parting in the source partition should move around the circle in the same direction as u but on the opposite side of the circle, while the parting in the target partitions should move in the opposite direction. Otherwise, edge lengths increase and with them the likelihood of crossings. Thus, starting with winding values $\psi_u(e) = 1$ and $\psi_v(e) = 1$ for all $e = \{u, v\} \in E(v)$, parting counters are iteratively moved around the circles and mostly decreased in the following way:

1. First try to improve the parting at V_i, i. e., iteratively, the value of ψ_u for the current parting edge is decreased and the parting moves counter-clockwise to the next edge until this parting can no longer be improved.
2. For edges whose source winding are changed in step one, there may be better target windings which cannot be found in step three because the value of ψ_j has to be increased, i. e., for each affected edge, the value of ψ_j for the edge is increased until no improvement is made any more.
3. Finally try to improve the parting for each target partition V_j separately, i. e., for each V_j, the value of ψ_j for the current parting edge is decreased and the parting moves clockwise to the next edge until this parting can not be improved any further.

After each update, it is ensured that all counters are valid and that winding values are never increased above 1 and below -1.

Based on the above, the locally optimal position of a single vertex can be found by iteratively swapping the vertex with its neighbor and updating the edge winding while keeping track of the change in crossing number. After the vertex has passed each position, it is placed where the intermediary crossing counts reached their minimum. Repositioning each vertex once in this way is called a *round of sifting*.

Theorem 9. *The running time of multi-circular sifting is in* $\mathcal{O}(|V||E|^2)$.

Proof. Computing the difference in cross-count after swapping two vertices requires $\mathcal{O}(|E|^2)$ running time for one round of sifting. For each edge, the winding changes only a constant number of times because values are bounded, source winding and target winding are decreased in steps one and three, respectively, and the target winding is only increased for edges whose source winding decreased before. Counting the crossings of an edge after changing its winding takes time $\mathcal{O}(|E|)$ in the worst-case. Actually, only edges incident to the at most two involved macro-vertices have to be considered. For each vertex $u \in V$, the windings are updated $\mathcal{O}(|V| \cdot \deg(u))$ times, once per position and once per shifted parting. For one round, this results in $\mathcal{O}(|V||E|)$ winding changes. Together, the running time is in $\mathcal{O}(|V||E|^2)$. □

3.6 Example: Email Communication Network

The strength of a multi-circular layoutis the coherent drawing of vertices and edges at two levels of detail. It reveals structural properties of the macro-graph and allows identification of micro-level connections at the same time. The showcase for the benefits of our micro/macro layout is an email communication network of a department of the Universität Karlsruhe (TH). The micro-graph consists of 442 anonymized department members and 2 201 edges representing at least one email communication in the considered time frame of five weeks. At the macro-level, a grouping into 16 institutes is given, resulting in 66 macro-edges. In the following drawings, members of the same institute are colored identically.

We start by inspecting drawings generated by a general force-directed approach similar to the method of Fruchterman and Reingold [26] and by multidimensional scaling (MDS) [17], see Fig. 10. Both methods tend to place adjacent vertices near each other but ignore the additional grouping information. Therefore, it is not surprising that the drawings do not show a geometric clustering and the macro-structure cannot be identified. Moreover, it is difficult or even impossible to follow edges since they massively overlap each other.

More tailored for the drawing of graphs with additional vertex grouping are the layout used by Krebs [37], and the force-directed attempts to assign vertex positions by Six and Tollis [50] and Krempel [38]. All three methods place the vertices of each group on circles inside of separated geometric areas. While some efforts are made to find good vertex positions on the circles, edges are simply drawn as straight lines. Figure 6(a) gives a prototypical example of this layout style. Although these methods feature a substantial progress compared to general layouts and macro-vertices are clearly visible, there is no representation of macro-edges and so the overall macro-structure is still not identifiable.

Finally, we investigate multi-circular visualizations of the email network. Its combinatorial descriptions allows for enrichments with analytical visualizations of the vertices. In Fig. 11 the angular width of the circular arc a vertex covers is proportional to its share of the total inter-partition edges of this group. The height from its chord to the center of the circle reflects the fraction of present to possible intra-edges.

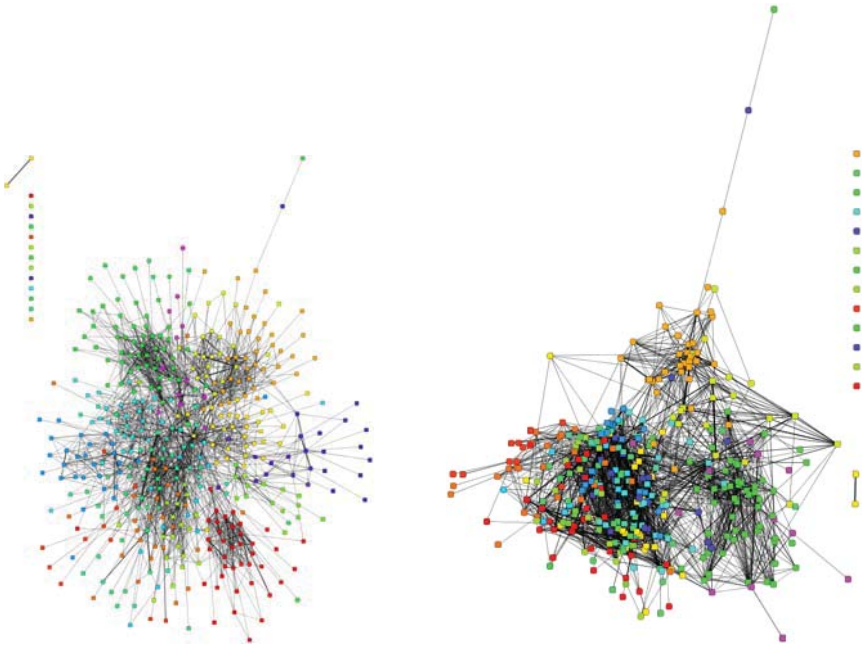
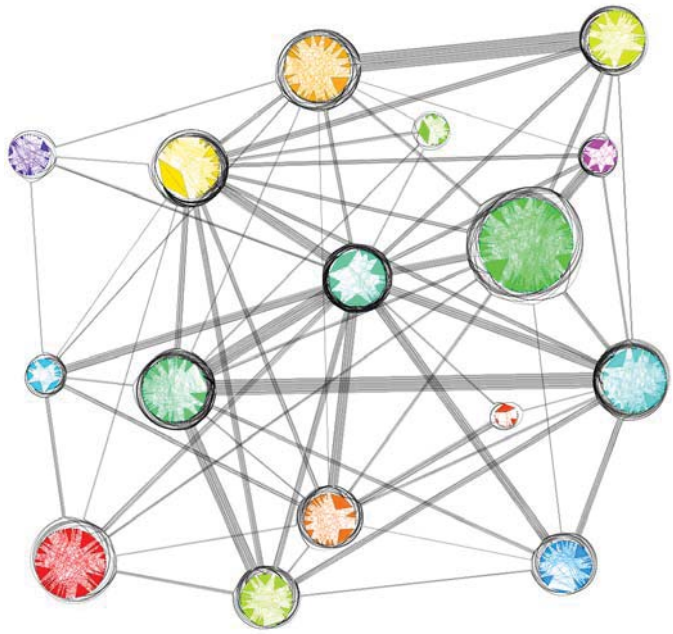


Fig. 10. Drawings of the email network generated by a force-directed method (left) and by multidimensional scaling (MDS, right). The colors of the vertices depict the affiliation to institutes.

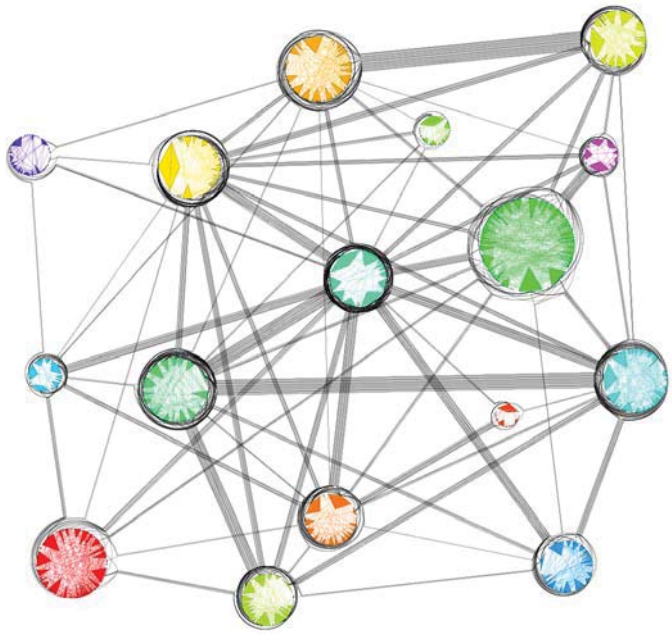
In order to investigate the effect of improved vertex orders and appropriate edge windings, we compare three variations of drawing heuristics for multi-circular layouts: shortest-path edge winding combined with random vertex placement and with barycenter vertex placement, and our multi-circular sifting (see Fig. 11). Through the grouping of micro-edges the macro-structure of the graph is apparent at first sight. A closer look reveals the drawback of random placement: edges between different groups have to cover a long distance around the vertex circles and are hard to follow. Also a lot of edge crossings are generated both inside of the groups and in the area around the vertex placement circles. Assigning vertex positions according to the barycenter heuristic results in a clearly visible improvement and allows the differentiation of some of the micro-edges. Using sifting improves the layout even further, resulting from a decrease of the number of crossings from more than 75 000 to 57 400 in the considered email network. The time for computing the layout of this quiet large graph is below 10 seconds.

3.7 Final Remarks

From the micro-layout algorithm we get a combinatorial description of the layout, i. e., circular vertex orders for each partition and edge windings, which allows



(a) barycenter (68 300 crossings)



(b) sifting (57 400 crossings)

Fig. 11. Multi-circular layouts of the email network

arbitrarily rotating each circular order without introducing new crossings (see Section 3.4). Therefore, for each partition, a rotation is chosen which minimizes the total angular span of the inter-partition edges, reserve space for the drawing of these edges, and place the vertices accordingly at a uniform distance from the border of the macro-vertex.

A major benefit of the multi-circular layout is its combinatorial description since it allows the combination with other visualization techniques to highlight some graph properties or to further improve the visual appearance.

4 Conclusion

In this paper we presented methods to analyze and visualize group-structure in social networks. The first part is focused on the definition and computation of structural network positions. We started by reviewing previous approaches for this task and distinguished them by two different criteria: first, whether the method establishes equivalence of actors or similarity of actors (discrete vs. real valued approaches) and, second, whether similarity is based on neighborhood identity or neighborhood equivalence. We then presented a novel framework for defining and computing network positions that is general and enjoys several methodological advantages over previous approaches. The second part of this paper introduced a visualization technique that shows the interplay between fine-grained (individual level) and coarse-grained (group level) network structure. Special emphasis has been given to the algorithmic optimization of esthetic criteria such as reducing the number of edge crossings.

References

1. Achlioptas, D., Fiat, A., Karlin, A., McSherry, F.: Web Search Via Hub Synthesis. In: Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2001), pp. 500–509 (2001)
2. Alon, N., Kahale, N.: A Spectral Technique for Coloring Random 3-Colorable Graphs. SIAM Journal on Computation 26, 1733–1748 (1997)
3. Archambault, D., Munzner, T., Auber, D.: TopoLayout: Multi-Level Graph Layout by Topological Features. IEEE Transactions on Visualization and Computer Graphics 13(2), 305–317 (2007)
4. Azar, Y., Fiat, A., Karlin, A., McSherry, F., Saia, J.: Spectral Analysis of Data. In: Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC), pp. 619–626 (2001)
5. Bachmaier, C.: A Radial Adaptation of the Sugiyama Framework for Visualizing Hierarchical Information. IEEE Transactions on Visualization and Computer Graphics 13(3), 585–594 (2007)
6. Balzer, M., Deussen, O.: Level-of-Detail Visualization of Clustered Graph Layouts. In: Asia-Pacific Symposium on Visualisation 2007, APVIS 2007 (2007)
7. Batagelj, V., Doreian, P., Ferligoj, A.: An Optimizational Approach to Regular Equivalence. Social Networks 14, 121–135 (1992)

8. Baur, M., Brandes, U.: Crossing Reduction in Circular Layouts. In: Hromkovič, J., Nagl, M., Westfechtel, B. (eds.) WG 2004. LNCS, vol. 3353, pp. 332–343. Springer, Heidelberg (2004)
9. Borgatti, S.P., Everett, M.G.: Notions of Position in Social Network Analysis. Sociological Methodology 22, 1–35 (1992)
10. Brandes, U.: On Variants of Shortest-Path Betweenness Centrality and their Generic Computation. Social Networks 30(2), 136–145 (2008)
11. Brandes, U., Erlebach, T. (eds.): Network Analysis: Methodological Foundations. Springer, Heidelberg (2005)
12. Brandes, U., Fleischer, D., Lerner, J.: Summarizing Dynamic Bipolar Conflict Structures. IEEE Transactions on Visualization and Computer Graphics, special issue on Visual Analytics 12(6), 1486–1499 (2006)
13. Brandes, U., Lerner, J.: Structural Similarity in Graphs. In: Fleischer, R., Trippen, G. (eds.) ISAAC 2004. LNCS, vol. 3341, pp. 184–195. Springer, Heidelberg (2004)
14. Brandes, U., Lerner, J.: Coloring Random 3-Colorable Graphs with Non-uniform Edge Probabilities. In: Královič, R., Urzyczyn, P. (eds.) MFCS 2006. LNCS, vol. 4162, pp. 202–213. Springer, Heidelberg (2006)
15. Brandes, U., Lerner, J.: Visualization of Conflict Networks. In: Kauffmann, M. (ed.) Building and Using Datasets on Armed Conflicts. NATO Science for Peace and Security Series E: Human and Societal Dynamics, vol. 36. IOS Press, Amsterdam (2008)
16. Coja-Oghlan, A.: A Spectral Heuristic for Bisecting Random Graphs. In: Proceeding of the 16th ACM-SIAM Symposium on Discrete Algorithms, pp. 850–859 (2005)
17. Cox, T.F., Cox, M.A.A.: Multidimensional Scaling, 2nd edn. Monographs on Statistics and Applied Probability. Chapman & Hall/CRC, Boca Raton (2001)
18. Cvetković, D.M., Doob, M., Sachs, H.: Spectra of Graphs. Johann Ambrosius Barth Verlag (1995)
19. Doreian, P., Batagelj, V., Ferligoj, A.: Generalized Blockmodeling. Structural Analysis in the Social Sciences, vol. 25. Cambridge University Press, Cambridge (2005)
20. Duncan, C.A., Efrat, A., Kobourov, S.G., Wenk, C.: Drawing with Fat Edges. In: Mutzel, P., Jünger, M., Leipert, S. (eds.) GD 2001. LNCS, vol. 2265, pp. 162–177. Springer, Heidelberg (2002)
21. Eades, P., Kelly, D.: Heuristics for Reducing Crossings in 2-Layered Networks. Ars Combinatoria 21(A), 89–98 (1986)
22. Everett, M.G., Borgatti, S.P.: Regular Equivalence: General Theory. Journal of Mathematical Sociology 18(1), 29–52 (1994)
23. Fiala, J., Paulusma, D.: The Computational Complexity of the Role Assignment Problem. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) ICALP 2003. LNCS, vol. 2719, pp. 817–828. Springer, Heidelberg (2003)
24. Fleischer, D.: Theory and Applications of the Laplacian. Ph.D thesis (2007)
25. Freeman, L.C.: Visualizing Social Networks. Journal of Social Structure 1(1) (2000)
26. Fruchterman, T.M.J., Reingold, E.M.: Graph Drawing by Force-Directed Placement. Software - Practice and Experience 21(11), 1129–1164 (1991)
27. Gaertler, M.: Algorithmic Aspects of Clustering – Theory, Experimental Evaluation and Applications in Network Analysis and Visualization. Ph.D thesis, Universität Karlsruhe (TH), Fakultät für Informatik (2007)
28. Gansner, E.R., Koren, Y.: Improved Circular Layouts. In: Kaufmann, M., Wagner, D. (eds.) GD 2006. LNCS, vol. 4372, pp. 386–398. Springer, Heidelberg (2007)
29. Gansner, E.R., North, S.C.: Improved Force-Directed Layouts. In: Whitesides, S.H. (ed.) GD 1998. LNCS, vol. 1547, pp. 364–373. Springer, Heidelberg (1999)

30. Gkantsidis, C., Mihail, M., Zegura, E.W.: Spectral Analysis of Internet Topologies. In: Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (Infocom), vol. 1, pp. 364–374. IEEE Computer Society Press, Los Alamitos (2003)
31. Godsil, C., Royle, G.: Algebraic Graph Theory. Graduate Texts in Mathematics. Springer, Heidelberg (2001)
32. Golub, G.H., van Loan, C.F.: Matrix Computations. John Hopkins University Press, Baltimore (1996)
33. Harel, D., Koren, Y.: Drawing Graphs with Non-Uniform Vertices. In: Proceedings of the 6th Working Conference on Advanced Visual Interfaces (AVI 2002), pp. 157–166. ACM Press, New York (2002)
34. Holten, D.: Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data. IEEE Transactions on Visualization and Computer Graphics 12(5), 741–748 (2006)
35. Kannan, R., Vempala, S., Vetta, A.: On clusterings: Good, Bad and Spectral. Journal of the ACM 51(3), 497–515 (2004)
36. Kaufmann, M., Wiese, R.: Maintaining the Mental Map for Circular Drawings. In: Goodrich, M.T., Kobourov, S.G. (eds.) GD 2002. LNCS, vol. 2528, pp. 12–22. Springer, Heidelberg (2002)
37. Krebs, V.E.: Visualizing Human Networks. Release 1.0, pp. 1–25 (1996)
38. Krempel, L.: Visualisierung komplexer Strukturen. Grundlagen der Darstellung mehrdimensionaler Netzwerke. Campus (2005)
39. Lerner, J.: Role Assignments. In: Brandes, U., Erlebach, T. (eds.) Network Analysis. LNCS, vol. 3418, pp. 216–252. Springer, Heidelberg (2005)
40. Lerner, J.: Structural Similarity of Vertices in Networks. Ph.D thesis (2007)
41. Lorrain, F., White, H.C.: Structural Equivalence of Individuals in Social Networks. Journal of Mathematical Sociology 1, 49–80 (1971)
42. Luczkovich, J.J., Borgatti, S.P., Johnson, J.C., Everett, M.G.: Defining and Measuring Trophic Role Similarity in Food Webs Using Regular Equivalence. Journal of Theoretical Biology 220(3), 303–321 (2003)
43. Masuda, S., Kashiwabara, T., Nakajima, K., Fujisawa, T.: On the $\mathcal{NP}$-Completeness of a Computer Network Layout Problem. In: Proceedings of the 20th IEEE International Symposium on Circuits and Systems 1987, pp. 292–295. IEEE Computer Society, Los Alamitos (1987)
44. Matuszewski, C., Schönfeld, R., Molitor, P.: Using Sifting for k-Layer Straightline Crossing Minimization. In: Kratochvíl, J. (ed.) GD 1999. LNCS, vol. 1731, pp. 217–224. Springer, Heidelberg (1999)
45. McSherry, F.: Spectral Partitioning of Random Graphs. In: Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2001), pp. 529–537 (2001)
46. Papadimitriou, C., Raghavan, P., Tamaki, H., Vempala, S.: Latent Semantic Indexing: A Probabilistic Analysis. Journal of Computer and System Sciences 61(2), 217–235 (2000)
47. Rudell, R.: Dynamic Variable Ordering for Ordered Binary Decision Diagrams. In: Proceedings of the 1993 IEEE/ACM International Conference on Computer-Aided Design, pp. 42–47. IEEE Computer Society Press, Los Alamitos (1993)
48. Sailer, L.D.: Structural Equivalence: Meaning and Definition, Computation and Application. Social Networks 1, 73–90 (1978)
49. Schrodt, P.A., Davis, S.G., Weddle, J.L.: Political Science: KEDS - A Program for the Machine Coding of Event Data. Social Science Computer Review 12(3), 561–588 (1994)

50. Six, J.M., Tollis, I.G.: A Framework for User-Grouped Circular Drawings. In: Liotta, G. (ed.) GD 2003. LNCS, vol. 2912, pp. 135–146. Springer, Heidelberg (2004)
51. Stewart, G.W., Sun, J.-G.: Matrix Perturbation Theory. Academic Press, London (1990)
52. Vempala, S., Wang, G.: A Spectral Algorithm for Learning Mixtures of Distributions. In: Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2002 (2002)
53. Wang, X., Miyamoto, I.: Generating Customized Layouts. In: Brandenburg, F.J. (ed.) GD 1995. LNCS, vol. 1027, pp. 504–515. Springer, Heidelberg (1996)
54. Wasserman, S., Faust, K.: Social Network Analysis: Methods and Applications. Cambridge University Press, Cambridge (1994)
55. White, D.R., Reitz, K.P.: Graph and Semigroup Homomorphisms on Networks of Relations. Social Networks 5, 193–234 (1983)