MatrixExplorer: a Dual-Representation System to Explore Social Networks

Nathalie Henry and Jean-Daniel Fekete

Abstract— MatrixExplorer is a network visualization system that uses two representations: node-link diagrams and matrices. Its design comes from a list of requirements formalized after several interviews and a participatory design session conducted with social science researchers. Although matrices are commonly used in social networks analysis, very few systems support the matrix-based representations to visualize and analyze networks.

MatrixExplorer provides several novel features to support the exploration of social networks with a matrix-based representation, in addition to the standard interactive filtering and clustering functions. It provides tools to reorder (layout) matrices, to annotate and compare findings across different layouts and find consensus among several clusterings. MatrixExplorer also supports Node-link diagram views which are familiar to most users and remain a convenient way to publish or communicate exploration results. Matrix and node-link representations are kept synchronized at all stages of the exploration process.

Index Terms— social networks visualization, node-link diagrams, matrix-based representations, exploratory process, matrix ordering, interactive clustering, consensus.

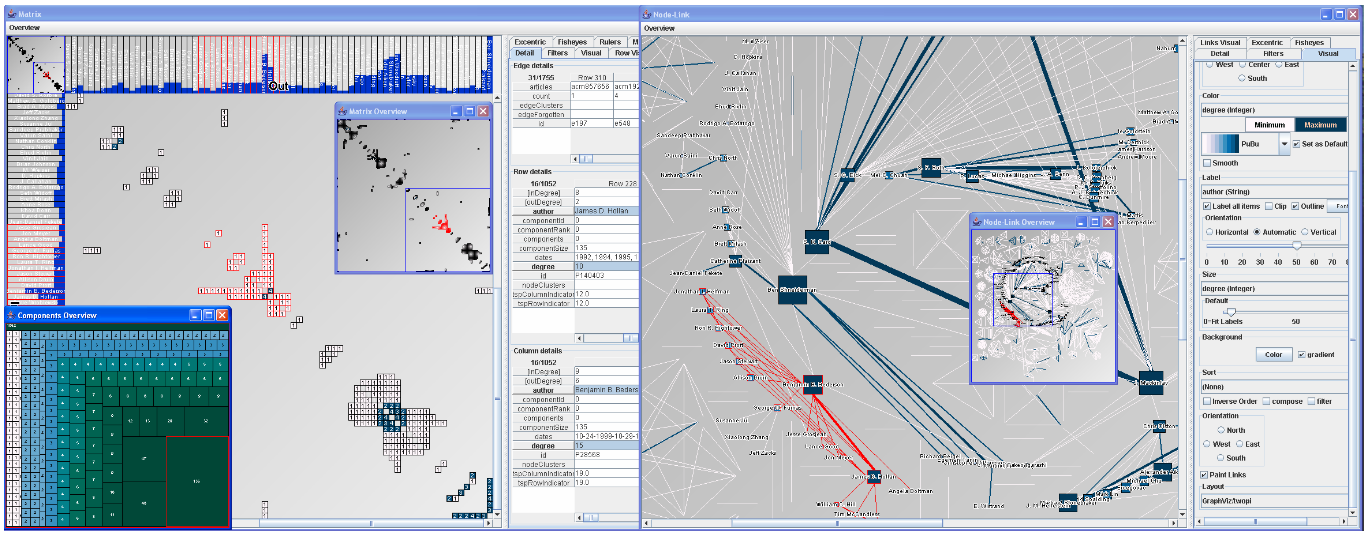


Fig. 1. MatrixExplorer showing two synchronized representations of the same network: matrix on the left and node-link on the right.

1 Introduction

Information visualization has been used to support social network analysis since the 1930s. Social scientists use visual representations both to explore datasets and to communicate their results. Some information visualization systems focus on exploration, taking advantage of features of the human perceptual system to discern visual patterns in the data. Others help researchers draw social networks, usually in the form of node-link diagrams to represent trees and graphs. Although adjacency matrices have played an important role in social networks analysis since the 1940s [16], few social scientists use their visual representations to communicate their findings.

This article presents MatrixExplorer (Figure 1), which offers both node-link and matrix representations to help sociologists and historians explore and communicate social networks. The node-link diagrams provide intuitive representations for relatively small networks, and, when adequately visualized, remain a powerful means of communication. MatrixExplorer also provides tools for reorganizing, clustering and filtering graphs using a matrix representation. These matrices are always readable, even for large and dense graphs, and thus support exploration throughout the analysis process. MatrixExplorer offers several novel features to help explore complex social networks, using the most suitable representation at any time.

This paper is organized as follows: we first present related work and describe the requirements for a visual exploration system that we defined together with social sciences researchers. We then describe MatrixExplorer and detail its major features for matrix-based representations. We conclude with discussion and future work.

- *Nathalie Henry is with INRIA Futurs/LRI and University of Sydney, E-Mail: Nathalie.henry@lri.fr.*
- *Jean-Daniel Fekete is INRIA Futurs/LRI, E-Mail: jean-daniel.fekete@inria.fr.*

Manuscript received 31 March 2006; accepted 1 August 2006; posted online 6 November 2006.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

2 RELATED WORK

Social networks are structures made of actors (generally humans) linked by relations. For example, these relations may be phone calls or kinship. Social networks are mostly trees or graphs; therefore social network visualization is closely related to graph visualization. A large amount of work has been done in the field of graph visualization; [22] presents a survey and the book of Battista et al. [4] is a good introduction. Several applications are dedicated to end users' visualization of social networks such as ContactMap [30], Vizster [20] or FlickrGraph. In this article, we focus on tools for professional social science researchers.

2.1 Social Networks Analysis and Visualization

A large number of systems exist[1] to visualize and analyze social networks and graphs in general. We can broadly define two categories: programming-based and menu-based systems.

Programming systems are highly tunable and provide a wide range of algorithms for visualization and analysis. Among the most used are JUNG [31] and GraphViz [17]. These systems contain most of the effective and efficient algorithms to draw trees or graphs. However, they are generic and – since they do not provide any particular support for social network exploration and analysis – they should be specialized using some kind of programming, far beyond the skills of social science researchers.

With menu-based systems, users can choose various algorithms from the file loading to the rendering, including the clustering or partitioning. They provide a basic interface but do not require programming knowledge. These systems contain dozens of algorithms and statistical measures; they answer almost all questions about social networks. The only problem is to know how to ask them. Expert users can parameterize each algorithm and end up with an effective visualization of their networks. For novice users however, the exploration boils down to a long trial and error process. The most popular free systems of this category include Pajek [10], UCINet [6], Tulip[3] and, for non-free systems: VisuaLinks [35] and Analyst's Notebook [24]. Pajek is certainly the most used in social sciences analysis. In practice, we observed one Pajek specialist – member of our participatory design group. He advised other researchers and gave them "recipes" in the form of a list of instructions to display their networks effectively.

Menu-based systems are not intuitive enough to be used for a primary exploration of networks since they are designed to support fine analysis conducted by an expert, not to support the discovery process.

GUESS [1] is a recent system oriented towards exploration. It contains visualization, statistical and analytics features as well as a script language. It only supports node-link representations but provides a relatively simple programming environment. It remains unclear whether social science researchers will invest time to learn the syntax and reach and adequate level of understanding. The same question arises for statistical packages where GUI-based system tend to attract more social science researchers than language-based packages like R [25], but both co-exist

2.2 Matrix-Based Representations

Bertin in "Semiology of graphics" [5] introduced visual matrices to represent networks. Ghoniem et al. [18] showed that matrices outperform node-link diagrams for large graphs or dense graphs in several low-level reading tasks, except path finding. Bertin showed that matrices can be used to exhibit high-level structures by finding good permutations of their rows and columns. Thus, he qualified matrices as "reorderable". Reordering rows and columns of an adjacency matrix is similar to computing the layout for a node-link diagram: finding a layout that reveals some structure in the data. Related works can be divided into two categories: automatic and interactive systems.

[1] *A list is maintained at http://www.insna.org/INSNA/soft_inf.html*

2.2.1 Automatic Ordering

Automatically reordering rows and columns of matrices is a well-known problem with a wide range of related works across various fields such as mathematics, biology and architecture. When considering adjacency matrices, the range is even broader because it is then related to linear algebra, graph seriation, as well as a long list of classical combinatorial optimization problems.

Matrix ordering algorithms try to optimize an objective function useful for some network related operation. Diaz et al. describes some of the most generic objective functions in [11]: Bandwidth, Minimum Linear Arrangement (MinLA), Cutwidth, Modified Cut, Vertex Separation, Sum Cut, Profile, Edge Bisection and Vertex Bisection. These algorithms find a linear order of the vertices of a graph that optimizes either a function of the edge length (the distance between the two vertices), or of the number of crossings of the edges. Exact solutions to these functions are all NP-complete but some have good polynomial time approximations. Among these functions, some have been used for matrix visualization. Reducing the bandwith is related to diagonalizing the matrix, a goal expressed by Bertin. It consists in finding an order that minimizes the maximum edge length. No polynomial time approximation exists for the exact algorithm so Siirtola and Mäkinen devised a set of heuristics [33] to find an approximate solution. The MinLA problem consists in finding an order minimizing the sum of all edge lengths. Simulated annealing [26] and spectral-sequencing algorithms [2] have been proposed to solve this problem. Koren and Harel [27] proposed a linear-time heuristic which is still the state-of-the-art. The MinLA problem has been investigated by Koren and Harel as a way to improve 2D graph layouts by separating the axes [28], among several other methods. They show results for 2D layouts, but not for matrices. A systematic analysis of the other objective functions applied to visual matrix ordering is yet to be made.

Reordering binary matrices for image compression, DNA sequencing and archeological dating have been successfully solved using spectral methods [2]. The optimal order is computed from the Laplacian matrix of the underlying graph and by using the row order of its eigenvector with the smallest non-zero eigenvalue. This result is related to spectral methods for node-link graph layout that also uses the eigenvectors with the smallest non-zero eigenvalues of the same graph. Whereas spectral methods are popular in 2D graph layout, to our knowledge, they have not been applied to matrix layout yet.

All the previous methods are usually defined for adjacency (binary) matrices. However, most of them can easily be generalized to weighted graphs.

The second category of algorithms aims at organizing the matrix in blocks. Automatic methods to compute block ordering emerged early [19]; [8] presents a panel of related works also named matrix partitioning or block clustering. Most of these methods are issued from the bioinformatics field. Bioinformatics researchers use matrix ordering algorithms to organize microarray data (heat maps presenting gene expressions per experimental conditions) in order to identify similar genes. In the field of social networks analysis, researchers aim at finding groups of similar actors and defining their roles; this is called positional analysis and identified in [23] as a main interest. Blockmodeling [13] intends to achieve this objective by decomposing the matrix in blocks of different shapes, using either clustering methods or equivalence relations defined on the graph. These methods focus on abstracting the matrix into higher level blocks, ignoring the details, whereas Bertin's claim is that the Matrix is useful because it shows both the details and the overall structure when correctly ordered.

Early work by Chauchat and Risson [7] has also investigated three classification methods to reorder Bertin's matrices: automatic classification, factorial analysis and hierarchical analysis. They have been used as a starting point by Bertin to manually reorder large matrices.

2.2.2 Automatic Ordering

Interactive tools such as InfoZoom [34] or TableLens [32] focus on table data and propose interactive methods to reorder one dimension of the table according to one attribute (one column). Users can then quickly identify correlated columns. However, the method is biased towards one dimension so users are unlikely to discover correlated rows. Moreover, reordering a matrix according to several attributes using only 1D sorting is long and tedious, as the user has to think backwards. To sort a matrix according to the names, then dates, then category, the user has to order first by category, then by dates and finally, by names.

In the field of graph drawing, some work has been conducted towards human-guided algorithms [12]. However, to our knowledge, no system supports assisted matrix reordering.

3 Exploratory Analysis Requirements

We used participatory design techniques describes by Mackay [29] to understand the needs of social science researchers. After several interviews, we organized a participatory design session with professional social science researchers, selected for their frequent use of social network analysis tools. The participants included: a sociologist, a psychologist, a social network analysis specialist, two historians and five computer science researchers in the fields of HCI and Information Visualization. We focused on three specific questions:

1. How would you like to create a social network?
2. How would you like to edit a created social network?
3. How would you like to explore an unknown social network?

The session was organized in four stages. First, we presented participants the state-of-the-art tools in the domain of social network analysis and a broad range of novel HCI and InfoVis techniques for interacting with graphs and data. We explicitly avoided guiding them towards specific design techniques or tools. In the second stage, they split into small groups and generated ideas in a brainstorming session, which were then ranked. In the third stage, participants captured their ideas by creating paper prototypes (Figure 2) and then filming what it would be like to interact with them[2]. In the last stage, we reviewed the ideas altogether and gathered the common and important ones. Summarizing the working sessions, we ended up with a list of requirements for social networks exploratory analysis.

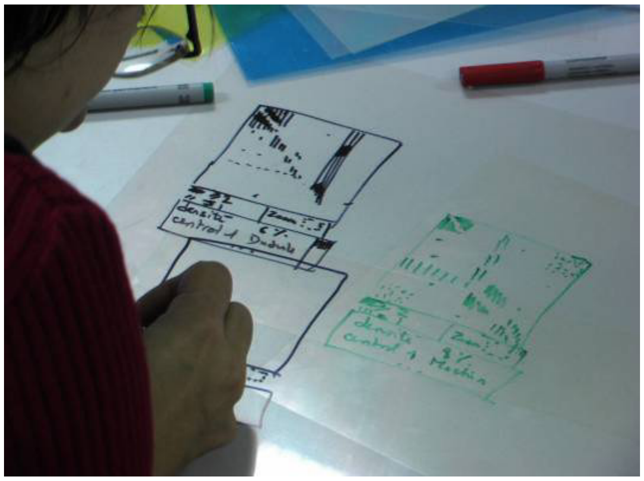


Fig. 2. Video Brainstorming showing a historian describing her ideas about using matrix-based representations to compare two networks.

[2] *See http://insitu.lri.fr/~nhenry/matrixExplorer/brainstorming/*

3.1 Requirements

R1 - Multiple representations: Participants used both node-link diagrams and matrix-based representations. Although node-link diagrams are familiar and effective to communicate (for relatively small or filtered-clustered graphs), they acknowledged that matrix-based representations were fast to display and easier to manipulate for large or dense graphs.

R2 - Connected components: Real graphs contain several connected components. Handling several connected components and being able to navigate within each of them, or compare them, is necessary for a system dealing with real datasets.

R3 - Overview: Overview is a challenge for large graphs. However, overviews are crucial for the exploratory process. They are used both as starting points for the exploration and as stable maps during the navigation. Overviews help users to build a mental map of their network. Participants asked for an overview of each visualization at all stages of the exploration.

R4 - Dataset general information: the type of graph, number of vertices, edges and, for each the number of attributes, their labels and types, should be displayed initially and be easy to access at any moment.

R5 - Attributes: Taking attributes into account makes the difference between graph drawing and information visualization. Participants were not interested in displaying a unique graph; they wanted to build several representations according to the different attributes of the edges and vertices. The structure of the graph may be different depending on the chosen attributes. Comparing these structures, understanding why they are similar or how they are different was their main concern. They need an information visualization system which helps them to choose visual variables for each attribute and create multiple views of their dataset. Consulting details for each vertex or edge was also a primary interest and therefore, details should always be visible or quickly accessible.

R6 - Analytical information: Visualizing and interacting with the data does not exclude statistical and analytical features. Participants wanted to get at least the basic network analysis data, such as number of actors, relations, density, diameter, five most connected actors, degree distribution. They also asked for computed attributes on demand, such as centrality measures.

R7 - Interaction vs. parameter tuning: Most of the participants were familiar with graph drawing and clustering algorithms. However, their understanding was limited and they were unable to finely tune the parameters for these algorithms. Thus, they asked for more interaction with the graph and less, or predefined, parameters. They also noticed that manipulating and reorganizing the network interactively facilitated understanding and memorization.

R8 - Layout: Computing the layout of a graph is necessary to find insights. It means computing coordinates for vertices in node-link diagrams or computing a permutation of rows and columns for matrices. In both cases, participants acknowledged that several layouts were required to understand a graph.
Participants asked for both automatic and manual (interactive) solutions. Automatic algorithms rarely provide satisfactory results but save users time and effort. Interaction let the user improve the resulting layout. Moreover, participants quoted that not being able to interact to drag a node or move a row or columns of a matrix was frustrating.

R9 - Filtering: For large networks, filtering is a requirement which allows fine analysis of the network and its sub-parts. However, filtering data may confuse users and lead them astray, especially if it alters the data structure. Therefore, participants asked that the system remind them that filtered data still exists.

R10 - Clusters: In social networks analysis, cluster detection or community detection is very important and required for exploration. A large panel of automatic methods exists to cluster networks. Users may also detect them manually. As for the graph layout, participants wanted to combine interaction with automatic algorithms. They expressed the need to handle several clusterings for a network and to annotate their clusters (giving each a name and a description).

R11 - Outliers: Social researchers are interested in outliers. For example, they try to understand why an actor has a different connection pattern or why two actors do not communicate within the same cluster. A system should not only filter outliers as dataset noise but support their discovery.

R12 - Consensus: Participants deal most of the time with multivariate data, i.e. several kind of relationships and attributes for actors. Thus they compute several clusterings depending on the attributes chosen and the visual representations. Participants asked for tools to identify a consensus among the clusters or differences.

R13 - Aggregation: Participants agreed that aggregating networks based on the clusters or communities was a useful feature to reduce the network size and help present results. However, they were concerned by the loss of information when dealing with aggregated networks and insisted in being able to get back to the full data when exploring the aggregated networks.

4 MatrixExplorer

MatrixExplorer is a first attempt to fulfil social sciences researchers' requirements for an exploratory system. In this section, we describe MatrixExplorer's main features.

4.1 Coupling node-link diagrams and matrices

MatrixExplorer is based on two representations: matrix-based and node-link diagrams meeting the first requirement of our users (*R1*). Node-link and matrix visualizations are synchronized in order to let the user work with both representations, switching smoothly from one to the other.

Multiple visualizations are synchronized by selection and filtering. Basically, if a user selects a set of actors in the matrix, this same set will be selected in all other visualizations (selection) and data filtered in one visualization will disappear from all others (filtering). Selection improves the transition from one representation to the other and constitutes the core of the coupling. Filtering preserves the coherence of the visualizations by presenting the same data, even if the attributes visualized are different.

In addition, visualizations can be synchronized by any visual attribute, simply by interactively setting the same attribute for the same visual variable. Thus, the user still has the possibility to not synchronize the visualizations in order to compare two attributes. With our system, users explore their networks using both representations, accomplishing tasks more easily with one representation or the other and visualizing the effect of a selection, or filtering, on all visualizations and their overviews.

Figure 3 shows a dual-representation of a co-authoring network and the correspondence of visual patterns in matrix and node-link representations. The process to obtain both representations follows: the user first automatically ordered the matrix, identified clusters (communities) and attributed colors to identify them. He then switched to a node-link diagram, displaying the community colors and laying the network out manually in order to better visualize how communities are linked and organized. Finally, moving back and forth between both representations, he identified the global structure of the network.

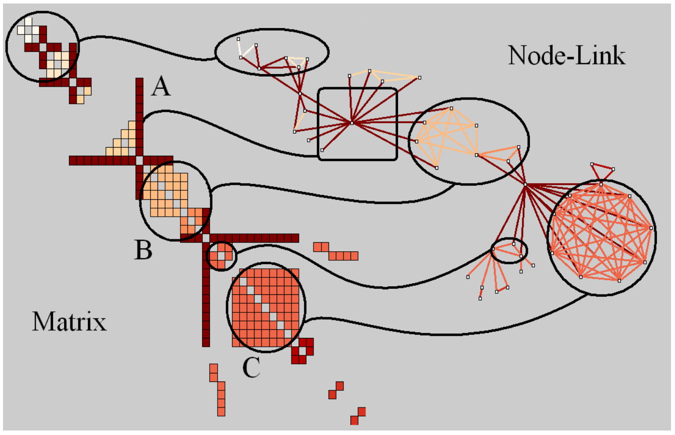
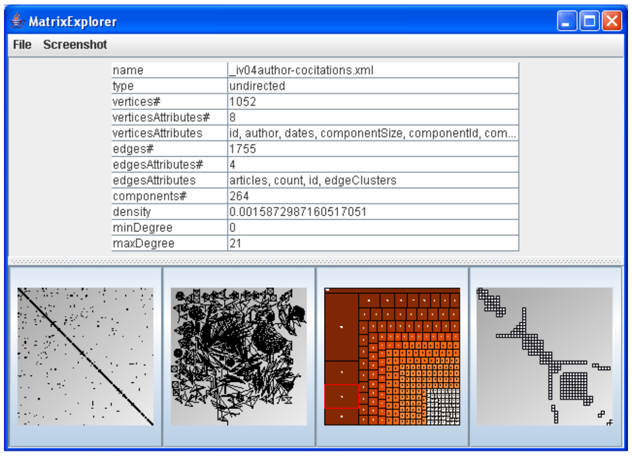


Fig. 3. Visual patterns in Matrix and Node-link representations of social networks. A represents an actor connecting several communities, B a community and C a clique (complete sub-graph).

4.2 Overviews

4.2.1 Datasets and workspace overview

MatrixExplorer proposes a quick overview of the user workspace. This overview includes for each dataset: general information on the graph and a visual overview of the related visualizations created by the user (Figure 4). This visualization covers requirements *R4* (dataset general information) and *R6* (analytical information) as well as *R3* (Overview).



name	_iv04author-cocitations.xml
type	undirected
vertices#	1052
verticesAttributes#	8
verticesAttributes	id, author, dates, componentSize, componentId, com...
edges#	1755
edgesAttributes#	4
edgesAttributes	articles, count, id, edgeClusters
components#	264
density	0.0015872987160517051
minDegree	0
maxDegree	21

Fig. 4. Workspace Overview. In this session, the user created 4 distinct visualizations (from left to right): matrix-based representation, node-link diagram of the full network, connected component visualization, and finally, matrix-based representation of one selected connected component.

Information appearing for each graph is: name, directed or not, number of vertices, number of edges, number of connected components, number of vertex attributes and their labels, the number of edge attributes and their labels, global density of the graph, minimum and maximum degree, and in/out degree if the graph is directed. We defined this list with social-science researchers, who specifically pointed out the lack of information about the attributed of vertices and edges (*R4* and *R6*).

The different visualizations related to a dataset are shown below this information, as thumbnails. With the workspace overview, the user has a reminder of existing visualizations, as exploring a social network can generate many windows and can show/hide visualizations in one click.

4.2.2 Connected components overview

The connected component visualization plays a special role in graph exploration. First, it is always a readable overview of the graph, quickly showing its macro-structure. Secondly, it is a starting point of the exploration as it filters matrices and node-link visualizations according to the current selected connected component. Social sciences researchers expressed this need in almost all interviews (*R2*).

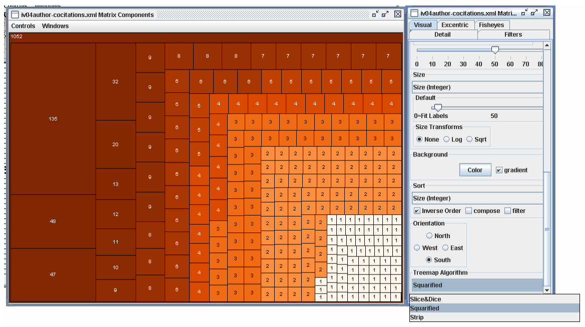


Fig. 5. Connected components visualization.

We choose to visually organize connected components as a compact rectangle in order to build a mental map of the macro-structure of the graphs (Figure 5). In this figure, the visual variables (rectangle size and color) are simply mapped to the number of vertices of the components (their size) and sorted by decreasing size from top left to bottom right. One click on a rectangle representing a connected component filters the synchronized visualizations. The user may map the visual variables to other attributes (using the control panel shown on the right); change the layout, or even the desired representation.

4.2.3 Visualizations overviews

In MatrixExplorer, an overview is provided for each visualization, meeting the requirement *R3*. The primary goal of an overview is to provide an overall picture of the visualization and thus help users identify the structure of the network and build a mental map.

In addition, we observed that users also use overviews as context reminders. While working on a matrix-based representation, they keep an eye on the node-link overview, to verify which part of the full graph they are working on. Moreover, they can directly observe the impact of their manipulations on the node-link diagram.

Finally, overviews are also navigation tools. A rectangle represents the current visualization's view. The user can grab it and move it to display a different part of the graph.

4.3 Visual variables

Assigning visual variables for each network attribute (*R5*) is a key to create effective visualizations. The InfoVis toolkit [15] provides the framework to interactively map attributes to visual variables for both node-link and matrix-based representations. Users are able to control the visualizations of actors choosing shape, size, color, texture and label and the relations (links) by choosing shape, length, color, thickness and label.

Useful interactions are also provided to favor direct manipulation (*R7*) and to improve the readability of the representations: a control panel lets the user assign each attribute to one or more visual variables, and dynamic filtering and sorting let the user choose what vertices are shown and in what order. Moreover, labels are often a main concern of social sciences researchers (*R5*). To show labels legibly on any visualization, The InfoVis Toolkit provides Excentric labeling and fisheye views.

4.4 Interactive layout

MatrixExplorer provides a number of graph drawing algorithms for node-link diagrams. They are mostly based on the JUNG and GraphViz packages. Essential interaction is also implemented such as moving nodes of the graph by clicking and dragging. In this section, we focus only on innovative features to manipulate matrix-based representations. We detail our interactive tools to reorganize their layouts (*R8*). We favored interaction and direct manipulation instead of iteratively adjusting a set of parameters (*R7*).

4.4.1 Basic interaction

MatrixExplorer provides a set of basic interaction tools essential, but not sufficient, for ordering large matrices. These tools include moving one or more rows or columns using drag and drop. They also include a feature (inspired by spreadsheet calculators) that allows users to sort rows and columns according to one attribute; for example, sorting rows according to the vertex names and columns according to the vertex degrees. Thus, rows can be used to find a specific vertex, and columns to find most/least connected vertices. Compared with InfoZoom of TableLens, the two dimensions can be ordered, and then be used to show the impact of one attribute on another one. MatrixExplorer also provides a tool to permute rows and columns circularly, similar to a "pan" tool in a paint program with the cells leaving on one side and entering on the opposite side.

4.4.2 Automatic ordering

As described in section 2.2.1, ordering algorithms focus either on ordering visual representations to let blocks emerge or on finding an optimal linear order for all the vertices of the graph. In MatrixExplorer, we propose to mix the two perspectives:

Block emerge if vertices directly linked in the graph are placed next to each other (consecutive ones in the matrix) – we propose to take into account a larger neighbourhood (distance>1) and to position vertices with similar connection patterns next to each other. To do this, we use the matrix of shortest paths (SP matrix) instead of the adjacency matrix. Our algorithm is:

```
Compute connected components
For each component
    Compute the SP matrix
Compute a matrix of distances between rows
            Apply the algorithm to find a linear order
            Compute a matrix of distances between columns
            Apply the algorithm to find a linear order
End for.
```

Connected components are independent blocks in the adjacency matrix so an order for their rows and columns is computed for each of them. Computing the SP matrix improves notably the order quality: it reduces the impact of noise (which is important in real datasets) and gives more information for low degree vertices (for which the rows and columns are very sparse). Computing the SP matrix is quadratic, as is the computation of the distance matrix for rows and the distance matrix for the columns. This has an important impact, since we want to use automatic ordering interactively. Therefore, we chose two fast ordering algorithms from the bioinformatics field. The first one is based on a hierarchical clustering, followed by a seriation (HCS) and is described in [14]; the second one is based on the traveling salesman problem (TSP) as presented in [9]. To solve TSP, we use a fast heuristic described in [21]. Matrices up to 1000 rows* 1000 columns can be ordered in seconds. Ordering larger matrices introduces a noticeable delay. So far, our user did not provide us with networks having a connected component larger than a thousand actors. However, we are investigating faster algorithms such as AMADO [7].

Both algorithms are based on a metric – similarity for HCS, distance for TSP – between the rows (respectively columns) of the matrix. Usually, this metric is either a Manhattan or Euclidian distance, or the Pearson correlation coefficient. By default, we choose a Manhattan distance.

Figure 6 presents a matrix reordered using TSP. The resulting matrix exhibits clearer blocks (diagonal with dense blocks); users can identify more clusters (*R10*) and articulation vertices between these clusters as dark color crosses here. A well-ordered matrix also helps identify outliers (*R11*) such as isolated relations, missing relation in a community, or actor with special connection patterns.

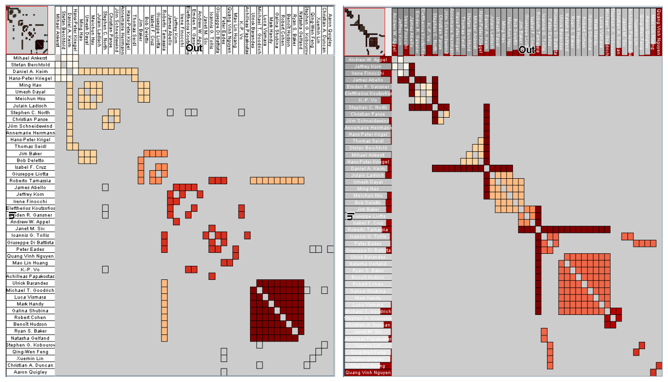


Fig. 6. Initial order (left) and TSP order (right). Colors represent clusters found by the user. Clusters are different in the two representations. Users found more clusters with TSP order. Headers red indicators (right) represents the distance between adjacent rows/columns.

4.4.3 Towards assisted ordering

In the second edition of "Semiology of graphics" [5], Bertin presents the results of three automatic ordering algorithms: automatic classification, factorial analysis and hierarchical analysis. He argued that none of these algorithms found a satisfactory matrix layout and performed some manual permutations to perfect them. We observed that automatic algorithms rarely provide a satisfying order for a given matrix and a user's taste. However, they save a substantial amount of time and effort and offer an initial layout which is better than a random one or a simple sort.

MatrixExplorer's goal is to propose a good initial matrix layout and to provide interactive tools to improve it, if needed. A "good" order, according to our participants, is one that reveals dense blocks and conversely avoids sparse isolated values. Sometimes, the initial layout – reflecting the data construction or collection method – is already good, as can be seen in Figure 6. This is the reason why we chose our TSP automatic algorithm which works by iteratively improving a given order.

Once a global layout is computed for the whole matrix, we propose that users interactively reorder sub-matrices they wish to explore. There are two options to adapt our automatic ordering algorithm to sub-matrices: 1) extract the sub-matrix from the initial SP matrix, or 2) compute the sub-graph corresponding to the sub-matrix selected by the user and then compute a new SP matrix.

The obvious drawback of computing a new sub-graph is the additional computations required: extraction of the sub-graph, computation of connected components and computation of SP matrix for each component. However, we observed that the number of vertices selected by the user is usually low, and thus the computation time is insignificant.

The second drawback has more impact on the user's understanding. The SP matrix computed for a given sub-graph contains notably less information than the initial SP matrix. Moreover, it may be misleading as the influence of all the unselected vertices is not taken into account. We implemented both solutions.

Figure 7 shows an example of the two sub-matrix ordering methods. We observed that results obtained with the first solution were more interesting as they let more blocks emerge as expected (SP matrix containing more information). This led us to favor the first version over the second. Figure 8 shows the sub-matrix reordered with the first solution, and the corresponding node-link diagrams.

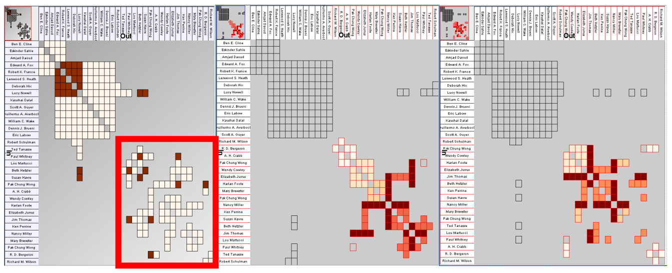


Fig. 7. In red sub-matrix to reorder. TSP order using the SP sub-matrix (middle), a new graph SP matrix (right).

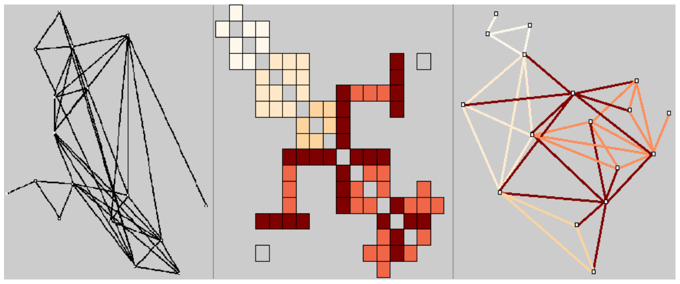


Fig. 8. a) Node-Link diagram with an initial layout using the twopi of GraphViz, b) sub-matrix reordered and colored by cluster interactively by the user, c) Node-Link diagram with a manual layout.

A main drawback of matrix-based representations is the 1D order of all vertices, which makes it difficult to represent articulation vertices between several clusters. However, well-ordered matrices let the user quickly identify communities and articulation vertices with a little training. Once communities are identified, the node-link diagram may be reorganized and clearly present the results. This is a major advantage of our dual-representation system: explore and discover with matrices, and present with node-link diagrams.

4.4.4 Locks

MatrixExplorer allows locking a set of rows and columns together before reordering a matrix. This functionality was not explicitly requested by our users but was detected during the use of our prototype. This feature is useful when, for example, a user identifies a community (set of actors) and wants to find out which external actors communicate with it. It is a constraint taken into account during the order computation: a single element is computed, representing the full set. The order computed takes into account this single element instead of the whole set. Our algorithm only keeps the first and last elements of the sequence and fills the distance between them in the SP matrix with a value of zero, to "glue" them together. The order is then optimized, as shown earlier, to integrate the sequence of elements, but these two are kept together. At the end, we insert the set back between the two elements.

4.4.5 Filtering or forgetting

We have implemented filtering in MatrixExplorer to fulfil the requirement *R9* to reduce the size and complexity of a network and finely analyze its sub-parts. Users can filter either actors or relations, according to one or a combination of all existing attributes (numerical, categorical or computed). In addition, to visualize the impact of an actor, or a set of relations on the network, MatrixExplorer provides a feature that "forgets" actors or relations and visualizes the resulting structure. This tool is slightly different from filtering: first, the element is still visible although it is made translucent; second, the changes in the new structure are highlighted to let the user rapidly identify the impact.

In Figure 9, the user identifies an actor collaborating with a community as well as a few external actors. He asks MatrixExplorer to forget all collaborations of this actor with the community by selecting it and visualizes the result.

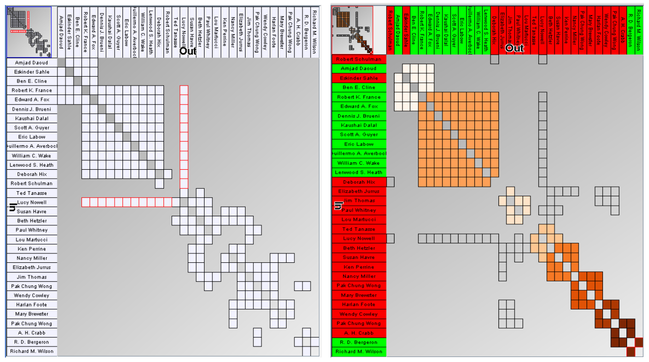


Fig. 9. Forgetting a number of collaboration for a key actor. Red headers indicate rows/columns with different neighbors, green indicates same neighbors. Other colors indicate communities identified by the user.

After the operation, the user can clearly see that the bottom right of the matrix is affected. The matrix is split up into two independent blocks: two sub-graphs. Thus, the forgotten actor identified is an articulation vertex (or a cut-point) between these two sub-graphs. Moreover, as several relations have been "forgotten", the lower sub-matrix has been reordered to let new blocks (communities colored in different nuance of brown) emerge, exhibiting the sub-graph structure.

4.5 Interactive clustering

The structure of the network changes, depending on the attribute or relation visualized. For example, two clusters of actors may be identified while visualizing a kinship relation, but three different clusters may be identified while visualizing their phone calls. Therefore, MatrixExplorer supports multiple clusterings (expressed as *R10*).

MatrixExplorer proposes two selection modes: click and drag or lasso. The lasso is used for fuzzy selection. Elements at the border of the lasso will be made translucent to denote the proportion that belongs to the selection. We implemented the fuzzy selection in response to users' observations. As matrices are very similar to tables or spreadsheets, users tend to adopt an "exact" or "precise" behavior when selecting groups. They spend considerable time determining whether a particular edge is included or not in a cluster. The lasso relaxes this behavior and exploration becomes more fluid.

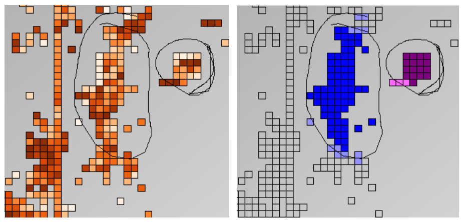


Fig. 10. Lasso selection on values visualization mode and resulting cluster visualization.

We observed that users also created clusters based on edge attributes – such as isocontours – and not only on blocks. We provided a tool to quickly switch from the standard visual mode showing colors based on an edge attribute, to colors based on cluster indices. These cluster indices are displayed in a transient mode, similar to vizster X-ray mode [20]. Users switch to this cluster visualization mode by pressing a key or a mouse button and switch back to the normal visualization mode by releasing the key or button. Users may also choose to display one clustering with a particular visual variable such as color or shape, since each clustering is implemented by a categorical attribute added to the edges.

4.6 Guiding the user: Finding a consensus

4.6.1 Consensus among layouts

Different layouts often imply different clusterings. It is important to be able to identify common clusters among layouts: i.e. to find a consensus when it exists (*R12*). MatrixExplorer offers this possibility. The procedure simply consists in identifying clusters as described in the previous section and ordering the matrix according to another layout. Clusters either explode in several parts or are conserved (Figure 11). The same method can be used to compare clusters of actors for different kind of relations.

In Figure 11, additional information on the clustering is displayed in the row and column headers. Depending on the algorithm used to reorder the matrix, this additional information is either a red histogram showing the distance between adjacent rows (columns) computed by TSP or the hierarchical clustering tree resulting of HCS presented as an icicle tree colored according to the similarity of its elements (blue and green in the Figure).

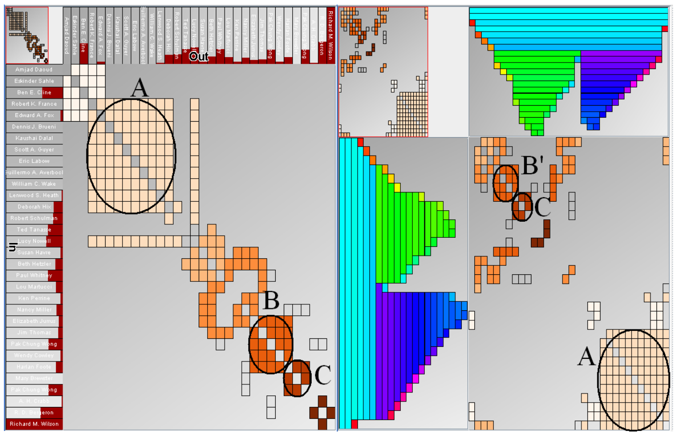


Fig. 11. Consensus between TSP (left) and HCS (right). We observed that a consensus exists for A, B and C. However, B is slightly different and lost some of its elements with HCS.

4.6.2 Consensus among clusterings

To find the differences between two clusterings, users have several options. To see global differences, users may switch from one clustering visualization mode to the other (X-ray) or simply choose a visual variable for one clustering and another for the second; for example, shape and color. We also provide a tool to precisely visualize the differences: users select two groups in each clustering and use the compare tool. Common elements are then displayed in green, and elements that appear in only one cluster in red.

5 Conclusion and Future Work

This article describes MatrixExplorer, a visualization system for exploring social networks for researchers in the social sciences. The system uses a dual-representation of social networks, the exploration process using mainly the matrix-based representation. We describe novel features designed to improve the usability of the matrix-based representations and propose a novel algorithm to reorder rows and columns of adjacency matrices that improve existing ordering methods designed for bioinformatics tables. We also described solutions to assist the user with interactive multi-dimensional reordering, interactive clustering and multiple clustering comparisons.

We designed MatrixExplorer with social researchers. We conducted several interviews and organized a participatory design session with sociologists, historians and social network analysts to formalize a list of requirements for a visual exploration system.

Our future work will include the extension of our ordering algorithm to directed graphs. In this case, the matrix of shortest paths between all vertices contains infinity values which are not supported by our current ordering algorithms. We also wish to improve the interactive clustering feature to support overlapping clusters and cluster hierarchies entry as well as cluster aggregation (requirement *R13*). We are also adding other ordering algorithms. To choose from these algorithms, we are studying which features could help users select the appropriate ones.

Finally, more work is still to be done concerning the coupling of visualizations. It is interesting to be able to synchronize and desynchronize visualizations (synchronizing on selection, filtering and datasets). Creating an intuitive interface to visualize and manage synchronizations without introducing too much complexity is a challenge.

This work will be available as an extension package of the InfoVis Toolkit.

Acknowledgements

We would like to thanks the participants of the design session for their feedback and Wendy Mackay for her support.
This work has been partly funded by the French RNRT Autograph Project.

References

[1] Adar, E., GUESS: A Language and Interface for Graph Exploration. in In Proceedings of ACM CHI 2006 Conference on Human Factors and Computing Systems, (Montréal, Canada, 2006), ACM Press, to be published.

[2] Atkins, J.E., Boman, E.G. and Hendrickson, B. A Spectral Algorithm for Seriation and the Consecutive Ones Problem. SIAM J. Comput., 28 (1). 297-310.

[3] Auber, D. Tulip: A huge graph visualization framework. in Mutzel, P. and Jünger, M. eds. Graph Drawing Softwares, Springer-Verlag, 2003, 105-126.

[4] Battista, G.D., Eades, P., Tamassia, R. and Tollis, I.G. Graph Drawing: Algorithms for the Visualization of Graphs. Prentice Hall PTR, 1998.

[5] Bertin, J. Sémiologie graphique : Les diagrammes - Les réseaux - Les cartes. Editions de l'Ecole des Hautes Etudes en Sciences, Paris, France, 1967.

[6] Borgatti, S., Everett, M. and Freeman, L. UCINET V user's guide. Analytic Technologies, Natick, MA, 1999.

[7] Chauchat, J.-H. and Risson, A., AMADO, a new method and a software integrating Jacques BERTIN's Graphics and Multidimensional Data Analysis Methods. in International Conference on VIsualization of Categorical Data, (Köln, R.F.A, 1995).

[8] Chen, C.-H., Hai-Gwo, H., Wen-Jung, J., Chiun-How, K., Yin-Jing, T., Sheng Li, T. and Han-Ming, W., Matrix visualization and information mining. in Proceedings in Computational Statistics 2004 (Compstat 2004), (Heidelberg, 2004), Physika Verlag, 85-100.

[9] Climer, S. and Zhang, W., Take a walk and cluster genes: a tsp-based approach to optimal rearrangement clustering. in Proceedings of the twenty-first international conference on Machine learning, (Banff, Alberta, Canada, 2004), ACM Press, 22.

[10] de Nooy, W., Mrvar, A. and Batagelj, V. Exploratory Social Network Analysis with Pajek. Cambridge University Press, 2005.

[11] Díaz, J., Petit, J. and Serna, M. A survey of graph layout problems. ACM Comput. Surv., 34 (3). 313-356.

[12] do Nascimento, H.A.D. and Eades, P., User Hints for Directed Graph Drawing. in Graph Drawing, 9th International Symposium, GD 2001, (Vienna, Austria, 2001), Springer-Verlag, 205-219.

[13] Doreian, P., Batagelj, V. and Ferligoj, A. Generalized Blockmodeling. Cambridge University Press, February 2005.

[14] Eisen, M.B., Spellman, P.T., Brown, P.O. and Botstein, D. Cluster Analysis and Display of Genome-Wide Expression Patterns. Proc. Natl. Acad. Sci. USA, 95. 14863-14868.

[15] Fekete, J.-D., The InfoVis Toolkit. in Proceedings of the IEEE Symposium on Information Visualization (INFOVIS'04), (2004), IEEE Computer Society, 167-174.

[16] Forsyth, E. and Katz, L. A matrix approach to the analysis of sociometric data, Preliminary report. Sociometry, 9. 340-347.

[17] Gansner, E.R. and North, S.C. An open graph visualization system and its applications to software engineering. Software - Practice and Experience, 30 (11). 1203-1233.

[18] Ghoniem, M., Fekete, J.-D. and Castagliola, P. On the readability of graphs using node-link and matrix-based representations: a controlled experiment and statistical analysis. Information Visualization, 4 (2). 114-135.

[19] Hartigan, J. Direct clustering of a data matrix. Journal of the American Statistical Association, 67 (337). 123-129.

[20] Heer, J. and Boyd, D. Vizster: Visualizing Online Social Networks. in Proceedings of the Proceedings of the 2005 IEEE Symposium on Information Visualization, IEEE Computer Society, 2005, 5.

[21] Helsgaun, K. An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic. European Journal of Operational Research, 126 (1). 106-130.

[22] Herman, I., Melançon, G. and Marshall, M.S. Graph Visualization and Navigation in Information Visualization: A Survey IEEE Transactions on Visualization and Computer Graphics 6 (1). 24-43

[23] Hummon, N. and Carley, K. Social networks as normal science. Soc Net, 15. 1-18.

[24] i2 Ltd. Analyst's Notebook. http://www.i2.co.uk.

[25] Ihaka, R. and Gentleman, R. R: A Language for Data Analysis and Graphics. Journal of Computational and Graphical Statistics, 5 (3). 299-314.

[26] Kirkpatrick, S., Gelatt, J. and Vecchi, M.P. Optimization by Simulated Annealing. Science, 220. 671-680.

[27] Koren, Y. and Harel, D. A Multi-scale Algorithm for the Linear Arrangement Problem. in WG '02: Revised Papers from the 28th International Workshop on Graph-Theoretic Concepts in Computer Science, Springer-Verlag, London, UK, 2002, 296--309.

[28] Koren, Y. and Harel, D. One-dimensional layout optimization, with applications to graph drawing by axis separation. Comput. Geom. Theory Appl., 32 (2). 115-138.

[29] Mackay, W., Using Video to Support Interaction Design. DVD Tutorial. in Proceedings of ACM conference on Human Factors in Computing Systems (CHI'02), (Minneapolis, MN, USA, 2002).

[30] Nardi, B.A., Whittaker, S., Isaacs, E., Creech, M., Johnson, J. and Hainsworth, J. Integrating communication and information through ContactMap. Commun. ACM, 45 (4). 89-95.

[31] O'Madadhain, J., Fisher, D., White, S. and Boey, Y.-B. The JUNG (Java Universal Network/Graph) Framework, University of California, Irvine, California, 2003.

[32] Rao, R. and Card, S.K. The table lens: merging graphical and symbolic representations in an interactive focus + context visualization for tabular information in Proceedings of the SIGCHI conference on Human factors in computing systems: celebrating interdependence ACM Press, Boston, Massachusetts, United States 1994 318-322

[33] Siirtola, H. and Mäkinen, E. Constructing and reconstructing the reorderable matrix. Information Visualization, 4 (1). 32-48.

[34] Spenke, M., Beilken, C. and Berlage, T. FOCUS: the interactive table for product comparison and selection in Proceedings of the 9th annual ACM symposium on User interface software and technology ACM Press, Seattle, Washington, United States 1996 41-50

[35] Visual Analytics Inc. VisuaLinks. http://www.visualanalytics.com.