ViAGraph: a Tool for Graph Visualization and Analysis

QUOC DINH TRUONG[1, 2]
TAOUFIQ DKAKI[1, 3]

1 GRIMM/ISYCOM Université Toulouse le Mirail5 Allée A. Machado 31058 Toulouse Cedex France
2 CanTho University College of Information Technology 1 rue Ly Tu Trong, CanTho VietNam
3 IRIT/SIG Institut de Recherche Informatique de Toulouse 118 route de Narbonne 31062 Toulouse France

Abstract

Graphs are common representations that can capture the structure and then can model a wide range of data and knowledge. In this paper, we present and discuss the functionalities of ViAGraph a tool for graph visualization and analysis. ViAGraph is meant to assist the user in exploring raw information in order to unveil interesting and useful information thru both query/answer and interactively guided data examination interactions. The paper presents a bunch of ideas and techniques related to graph visualization and exploration. Our main contributions are: 1. We propose a new approach of node placement based on 'geographic' constraints. 2. We discuss a novel analysis method based on graph comparison. Strengths and weaknesses of the proposed methods are discussed.

1. Introduction

Visualization is often an important key factor of success and added value enhancer of information management softwares as most of these softwares somehow and to some extent use similar techniques. There are several categories of visual representation that are frequently used; each of which is suitable for the visualization of specific type of information. Among these categories, graphs offer some valuable advantages with their simplicity in statement and representation, their high semantic convey capabilities, their suitability for practical and theoretical reasoning… Moreover, our special interest in graph visualization lies within the fact that graphs are common representations that can capture the structure and can model wide ranges of data and knowledge. Graphs and networks are good ways of describing and modeling systems composed of some objects –conceivably of several types- having some kinds of relationships –also possibly of several types-. Nevertheless, even when using good and convenient data representations such as graphs and networks, good softwares for information visualization must fully take advantage of these representations by providing efficient tools for data manipulation and embedded-information discovery.

In this paper, we present and discuss the functionalities of ViAGraph a tool for graph visualization and analysis. The purpose of ViAGraph is to assist the user to explore raw information and unveil interesting and useful information thru both query/answer and guided interactive data examination. This work is motivated by the fact that visualization, as part of Human-Computer interaction, is an important aspect in information and knowledge management. Its main goal is to help to, rapidly, grasp the meaning of either raw or elaborate information. We particularly focus on the visualization of capitalist and organizational networks where it is important to uncover invisible colleges and confront them to institutional information as the ones in organizational charts. Such cases stress the need of finding ways to deal with labeled and multipartite graphs where vertices and edges can

both be of several types. This includes handling visual representations in a multi viewpoints and multi-abstraction levels basis.

An important aspect of information visualization is data transformation to meet spatial-representations prerequisites (spatial coordinates …). This process of data pretreatment is carried out under some criteria optimization which main purpose is to lead to a useful visualization that enhances the perception of embedded meaning in data. The most common criterion in graph visualization is the crossing minimization [1]

As we mentioned earlier, we are mainly interested in networks visualization. We specially focus on multipartite-labeled graphs. Since then, we seek representation methods that help to better distinguish the different types of both vertices and edges. To do so, we use shapes, colors … to discriminate the different types of vertices and edges. ViAGraph also offers miscellaneous ad-hoc filtering techniques that help to focus on special types of nodes or relationships. Our main contribution is a model for over-constraining nodes space-localization in graph visualization. We argue that this model enhances networks analysis. The constraints are generally related to some meta-information or extra-network features; for example adding geographic constraints leads to a better understanding of geographic influences over inter-node relationships. Another example is 1D or 2D grid constraint used to evaluate the influences of one or two nodes classifications over the relationships in a given network.

In addition, dealing with large graphs underlines the need of reduction methods. In ViAGraph, we focus on data reduction rather than visualization-space enlargement. In ViAGraph, we propose some classification methods to reduce the complexity of graph visualization and to get handy, useful and easily interpretable multi-viewpoints representations.

Next sections of this paper are organized as follows. Section 2, exposes the data structures we use to represent graphs. Section 3 discusses the choice of visualization techniques. Special focus will be given to our over constrained Graph visualization approach. Section 4 introduces some cluster visualization methods for large networks visualization. Section 5 presents networks specific analysis techniques and relates some of our previews work [2], [3], [4] .The last section gives brief conclusions and perspectives.

2. Data structure

To fully define a graph one must provide information about all its vertices –typically IDs- and all its edges. Usually, these information are represented in two separate text files: one for vertices and the other for arcs. The most used formats are ASCII matrix, Ucinet and KrackPlot [4]. Thus, several different ways of representing graphs exist; each has objective and subjective advantages and disadvantages. In this section we present a personalized data structure to represent and store graphs. The first objective of this representation is to make it possible to easily and comprehensively take into account the possibility of having multiple types of both vertices ad edges. In ViAGraph, we are obviously interested in 'technical' graph drawing problems which main concern is to give the 'best picture'. However, we are more interested in visualization as part of the data analysis and mining processes which main goal is to uncover structure and characteristics of graphs. Efficiency of data analysis methods is enhanced by efficient access to raw information and extra-graph information. This gives the opportunity to cross-assess the findings of graph exploration.

In ViAGraph we use four files to represent and store graphs and related information –ideally this could be a queryable relational database. The files include information about vertices, visualization space, vertices localization in the visualization space, edges, and extra-network attributes such as classes and geographic information of both vertices and edges.

Figure 1 shows the structure of vertices file. It encompasses vertices labels and ID numbers. Figure 2 shows the structure of edges file. This file contains for every vertex a list of adjacent vertices along with other information about edges types and weight. Figure 3 gives extra information about vertices, defines types of edges and vertices. Extra-network information file is accessible at request

(querying) during the visualization and exploration stage. The three files permit the handling of graphs and networks having several types of both nodes and edges.

Fig. 1: Structure of nodes file

```
1   Label of vertex 1
2   Label of vertex 2
        …
n   Label of vertex n
```

Fig. 2: Structure edges file

```
n
m   type of rel     wt of rel
p   type of rel     wt of rel
…
```

Fig. 3 Structure of extra-network information file

*node info	2 Electricite	4 6 11 20 26
4	3 Eau	5 4 7 8 9 10 12 13 14 15 17 18 19 21 23 25 27 28
Nom R.Mirando Robredo	4 Autres	*tie info
Sexe Masculin	5 Administrateurs	1 Conseil d'administration
Add Toulouse	*node type	2 Direction
…	1 16 22 29	3 Conseil économique
*type info	2 1 2 3 24	
1 Gaz	3 5	

Fig. 4 : Structure of drawing file

#nodes coordinates	… …	#edges types
56 551	#is multipartite	-65536
637 504	true	-13395712
662 445	#nodes types	-65536
… …	2	-52225
#edges coordinates and sapes	1	-1
467 410 394 461	1	-65536
467 410 425 367	2	-6711040
199 425 198 461 394 461	2	-16763956
… …	…	…

Another important point besides obtaining graph representation is to safeguard this representation for later use in order to allow the user to resume his work from where he stopped it. Figure 4 gives an example of the graph-layout storage which includes the values of all visual variables used for graph redrawing.

3. Visualization techniques

3.1. Drawing algorithm

Usually, a small graph can be drawn manually so that the obtaining picture best shows the underlying relationships. With large graphs, this task becomes a more difficult work to achieve. Researchers have been competing, for years, to produce the ideal algorithm for automatic graph drawing. One among the most successful available methods is Force-directed placement method whose principle is to view a graph as a mechanical system of spring interactions. The original method is due to Eades [5] but his implementation did not reflect Hooke's law[1], rather, he chose a

[1] Hooke's law is a macroscopic approximation of the behavior of springs.

personal formula for the forces exerted by the springs. The most famous variant on Eades's algorithm is Kamada and Kawai's method [6]. Eades abandoned Hooke's law whereas Kamada and Kawai solved partial differential equations based on it to optimize the drawing. Kamada and Kawai's algorithm adds the concept of ideal distance between vertices that are not neighbors: *proportional to the length of the shortest path between them*. All Force-directed placement methods involve solving optimization problems, which, in turn, highlights the need for careful choice of "threshold values". This is certainly a difficult problem that affects the efficiency and usefulness of resulting drawing. In Eades's method the "threshold value" is directly related to the number of iterations whereas in Kamada and Kawai's method it has to do with the springs system total-energy.

We first tried the Eades's algorithm adding the concept of force used in [7]. This method performs well and we obtained good results in comparison with other works. Unfortunately the method was quite slow even with relatively small graphs. It takes several seconds for a graph that has few hundreds vertices and arcs to be drawn.

Fig. 5 GRIP Algorithm

ViAGraph is untended to deal with graph, which usually has over than hundreds of vertices and a big number of arcs. Consequently, the algorithm we first developed was not very satisfactory. As we mentioned before, it becomes slow when the number of vertices and arcs grows big. It is for this reason why we decided to shift to a more efficient and suitable method: namely Graph dRrawing with Intelligent Placement (GRIP) [8]. Fig 5 shows the principles of GRIP, which consists of three main phases. In the first phase, GRIP creates a maximal independent set (MIS) filtration V: $V = V_0 \supset V_1 \supset \ldots \supset V_k$ so that $k = O(\log n)$ where n is the total number of vertices, and $|V_k| = 3$; In each step beginning from step k, algorithm will determine vertices position in V_i thanks to the positions of vertices in V_{i+1}, in other words, for each step, we seek to position only vertices that are not in V_{i+1}. First, positions of these vertices are placed initially according to the positions of their closest nodes (GRIP second phase). Then, these positions are refined as consequences of attraction and repulsion forces (GRIP third phase). These two last phases are repeated until step V_0 is reached.

3.2. Visualization improvement

In this part, we briefly present selected tools and techniques for the purpose of visualization improvement.

In ViAGraph, arcs are represented as sets of adjacent line segments or polylines rather than simple straight lines. This enhances the capability reducing the number of crossings and leads to better visualizations. The process of shifting from a single straight line representation to a polyline is manual. Lines segments are interactively created and repositioned by a 'click, drag and drop' approach.

ViAGraph also offers several traditional rendering tools to help users to better examine graphs. Among these tools are rotations, zooms, thresholding, repositioning, recoloring … By using these operators, users can change layout of graphs to meet their point of views and can, as we mentioned in the introduction, better handle graphs with multiple types of vertices and arcs.

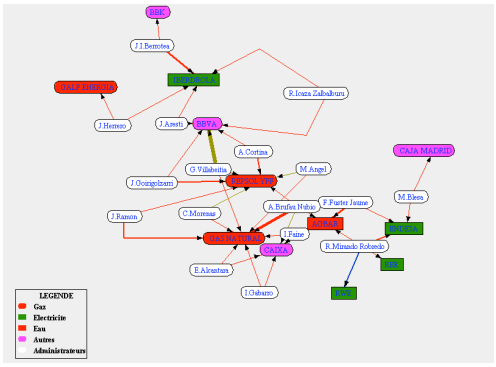
Another manner to increase the power of graph and network exploration thru visualization is to help the users to better appreciate the positions and the relative importance of both arcs and nodes that constitute the network. In ViAGraph, this is done by using the notion of metric values [9] [10].

F x = –k(x – x 0) = –k x

Metric values are numerical quantities associated to vertices or arcs and computed on the basis of graph structure or extra-graph information. Each vertex (arc) and each edge can be linked to an individual metric value. These values are graphically represented by means of several visual attributes (colors, shapes, size of shapes…). Several metric values can be simultaneously visualized. Each kind of metric values can be associated to a unique and different visual representation. They also can be summarized and represented as a unique value.

In case of weighted graphs, weight values can be considered as metric value associated with arcs. In such situation ViAGraph offers several approaches for arcs representation such as the labeling with weight values; the use of line characteristics as thickness, color and style. Since there is no absolute best representation for arc weights, it is up to the user to make his choice. The suitability of each choice depends on users' point of views. Visualizing directly weight values makes drawing more complex but users can easily recognize and rank these values while using color or thickness for coding weight values is better for drawing concern but users can not know the exact values.

Fig. 6: Using metric value for vertices and arcs



3.3. Visualization with constraints

In this sub-section, we address our probably most important contribution in this paper. As we mentioned earlier in this section most approaches of graph drawing are based on physical interactions –attraction and repulsion in the case of force based algorithms- involving nodes and edges. Our approach combines two different kinds of interaction: one between graphs basic components –nodes and edges- and the other between sub-graphs.

The visualization space is subdivided into non-overlapping –and, at least for the moment, convex- areas. Each graph vertex is assigned to a given area. This method restrains the drawing of specific nodes to specific related areas. Thus, visualization space is divided into areas with respect to some extra-network node attributes such as nodes' originated countries or nodes' categories. This approach facilitates the appreciation of correlations between network properties and extra network information.

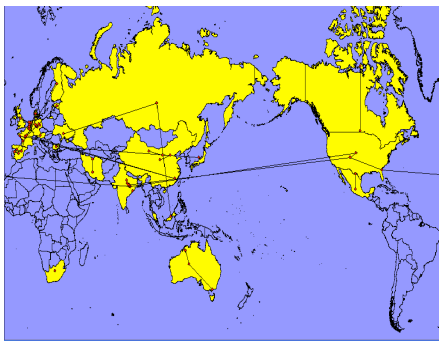
The overall goal of this 'geographic'-like constrained graph drawing and visualization is to place vertices sharing some common characteristics –e.g. geographic features- into a distinctive area. The assumption we made is that good representations tend not only to minimize the edges crossing but also to group together nodes having the same characteristics. We presuppose that such representation will help to better apprehend the intricating intra and inter group relations and position in the network. This could lead to uncover key roles and key actors in networks.

The process of over constrained placement is under taken in two phases:

- We first draw sub-graphs in related areas. The drawing is conducted according to the Graph dRrawing with Intelligent Placement algorithm. Sub-graphs induced by vertices of the same group are made connected beforehand to allow the use GRIP method,
- Then, we add the remaining arcs after computing the best placement the set of areas –this is possible only if relative position of areas is not mandatory as in geographic maps.

By adopting this method we consider that local drawings are more important than global drawing. Accordingly, vertices coordinates are computed in two phases: first, each sub-graph is created (by selecting its vertices and arcs) and drawn in a virtual space. Second, these sub-graphs are properly projected into private localization under crossing minimization criterion and using individual zoom coefficient.

Fig. 7 Graph visualized on world map



In case of graph representation on geographic map (fig 7), the domain of geographic suitable positions of each vertex is clearly identified. This has obvious advantages. Nevertheless, localization and size of each country being fixed, induced representation with over constrained drawing approach may lead to visualization presenting prohibitive edge crossings. In fact, there may be many vertices in small localizations whereas several big localizations may contain little vertices or no vertex at all. Although we support interactive tool for visualization refinement –e.g. local rotation, homothety …- which help to reduce the overall number of crossings, representations still can be far from perfect. This is why, even in the case of geographic maps, we provide non realistic but more convenient representations. These representations are originally designed to handle non-geographic constraints especially –but not solely – when they are related to some clustering method outcomes.

Fig. 8a Example of "non geographic" constraints

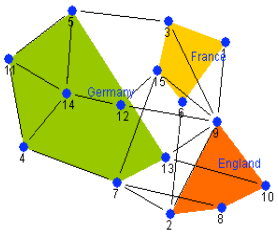
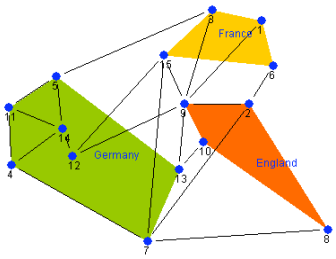


Fig. 8b Enhanced representation of graph in fig 8a



In case of constrained representations using non-geographic maps, the placement approach is slightly different since the contours of the disjunctive areas linked to the different groups of vertices must be created beforehand. These areas are expected to be convex for simple visualization results.

Figure 8a shows the result of our method of vertices constrained placement. Vertices of same group are placed in a same zone differentiated by its color.

ViAGraph supports tools that allow the user to modify representation in the context of constrained visualization. ViAGraph provides whole areas handling by mean of repositioning, rotation and zoom Figure 8b shows some changes made beginning from figure 8a and based on rotation and repositioning. Using localized zooms, users can increase size of area related to a given group and decrease size of the others zones –in a fish eye like approach- to focus on one group intra relationships without loosing the global context. The size decreasing can be conducted until entire collapse of manipulated areas.

4. IV Clustering visualization

A key issue of graph visualization is the size of graphs. The drawing algorithm described in the precedent sections gives quite good results. Unfortunately, when dealing with large graphs consisting of more than several hundred of nodes and edges, traditional difficulties appear. These difficulties concern the crucial performance issue especially in the data exploration systems where interactivity is an essential feature. They also concern usability and viewabiliy when the physical limits of viewing devices –screens- are reached. It becomes hard or even impossible to discern between modes and edges. This problem is well described in scientific literature. Many solutions are suggested among which one can find scrolling virtual viewing spaces, fisheye visualization, data reduction… In the past, we explored many of those solutions. There is no perfect solution but there are solutions that are suitable for exploratory data-analysis where combining local and global examination can be crucial. Among these solutions, ViAGraph offers interactive filtering methods, radial tree layout approaches [11] and multi-level abstraction visualization by mean of recursive agglomeration/decomposition.

In ViAGraph radial tree layout approach allows user to visualize nodes that are neighbors of a giving node –*the focus*. These neighbors are chosen following two limitations:

- Limit on radius. All lengths of paths between displayed nodes and the focus must be less than the radius. A vertex is said to be a first level neighbor if and only if it has a direct link to the focus.
- Limit on arcs weight: a path is considered as valid if all its edges have weights exceeding a giving threshold.

When dealing with large amount of data, one usually encounters overwhelming complexity problems. Complexity reduction becomes the key factor of success in interactive analysis processes. Cluster analysis methods have the ability to achieve this complexity reduction goal. Data is divided into several subsets to gain insight into it by giving the possibility to examine and analyze it at the clusters level. Clusters can be in their turn recursively divided until suitable size for convenient and easy analysis is reached.

Clustering methods are among the most used solutions for visualizing and analyzing large graphs. They allow changing point of views from global –inter cluster relationships- to local –intra cluster. Both internal graph structure and extra-graph information associated with vertices can be used for grouping purposes – they can be used as sources for computing similarities/dissimilarity between vertices.

Several methods for cluster analysis have been implemented in ViAGraph among which are : Hall [12], KCL [13] MCL [14], GSOM [15] and HAC [16].

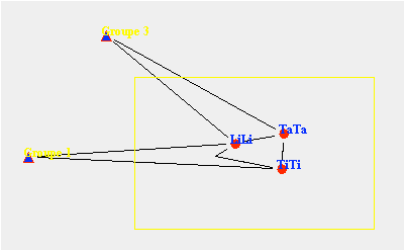
Each algorithm is based on a different approach, so that it gives a different view of the studied graph or network. We used the cluster algorithm of Hall [12] and that of Kernighan & Lin [13] as supervised algorithms. These algorithms are called bi-cluster algorithms since they construct two clusters from an original cluster by splitting it into two clusters and by dispatching vertices into them.

MCL algorithm (Markov clustering) [14] and GSOM (Growing Self Organizing Map) [15] are two examples of unsupervised algorithms. The MCL algorithm - Markov Cluster Algorithm-, is a fast and scalable unsupervised cluster algorithm for graphs based on simulation of stochastic flow in graphs. The MCL algorithm simulates flow using two algebraic operations on matrices. Its formulation is simple and elegant. GSOM -Growing Self Organizing Map- is based on a different approach. Vectors called model vectors are created and vector represent each a cluster. A cluster is made of vertices, which are closer to its model vector than to all the other vectors. The number of clusters generated by GSOM algorithm depends on desired the ratio between final and starting *'total error'* [15].

We also take into account a simple hierarchic cluster algorithm CAH [16]. This algorithm is based on a simple agglomerative approach. First, each vertex is viewed as a single cluster, then at each step, the two closest clusters –closeness is evaluated by using similarity functions or distance

measures over the node set e.g. Euclidean distance- are merged to construct a new group that replace them. There are a lot of ways to compute a distance between two clusters. Distance between two groups can be the maximum/ minimum/ average of all distance between the vertices of the two groups. Algorithm runs until all vertices belong to a single cluster.

Fig. 9 folded an unfolded cluster



In graph visualization, cluster analysis solves the size problem but it hides the complete structure of graphs which is a serious drawback of this approach. It is valuable to handle data at the same moment at both the clusters level initial vertices level. By unfolding or shrinking a cluster, ViAGraph allows the user to visualize several clusters at the same moment. Figure 9 shows vertices and clusters visualization. The vertices belonging to a same unfolded cluster are mandatory kept close together. They can only move inside a limited but resizable zone indicated by yellow rectangle. Users can also consider one cluster as a novel graph by visualizing it in new window.

5. *Analysis technique*

Analysis techniques presented in this section allow users to understand the network and the role each node plays. The analysis results are, at will, added in the visualization or saved in a text file.

5.1. *Centrality*

Measures of centrality help to determinate the relative roles of the nodes of a given network [4]. There are four basic measures of centrality: Degree, closeness, betweenness and eigenvalue.

- *Degree centrality*: the degree centrality of a node is measured as the number of its ties. In directed network, we distinguish two types of degree: in-degree and out-degree.
- *Closeness centrality*: contrary to centrality, closeness takes into account nodes' indirect ties. It computes the average path lengths separating a given node from the others.
- *Betweenness centrality*: betweenness indicates how often a node finds itself in the shortest path between two nodes.
- *Eigenvalue centrality*: eigenvalue is an important centrality measure which is used in many applications. In social networks, we can have two actors with the same degree centrality but one could know only actors who hardly know few others while the second knows actors who have big degree centrality. In this case, degree centrality does not distinguish who is most important among these two actors whereas eigenvalue centrality does.

5.2. *Other analysis - Structural equivalence*

ViAGraph offers ways to determine and represent structural equivalence measures between two given nodes. These measures [3] help identifying the nodes that are structurally equivalent –who play the same role in a given network. Structural equivalence can be computed according to several approaches.

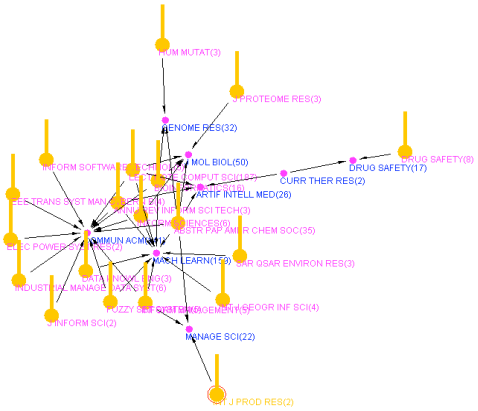
- *Exact matches*: It counts the number of shared neighbors by a couple of nodes.

- *Pearson correlation coefficient*: In case of valued or oriented graphs, exact matches measure lacks the ability to convey the essence of structural equivalence concept. On the contrary, Pearson correlation coefficient can be used. For two given nodes, it is computed as the correlation between the vectors associated to the nodes and derived from the graph incidence matrix [3].
- *Euclidean distance*: This measure computes the dissimilarity between two nodes. It can be used with either valued or binary data.
- *Jaccard's coeffi*cient: In case of sparse networks where density is low Jaccard's coefficient is used [3]. It has better ability to distinguish structural equivalence in comparison to previously mentioned measures.

5.3. Graph comparison

ViAGraph implements a novel method we proposed in [2] to compare vertices of a graph with vertices of another Graph (considered as a model). Let G_1 and G_2 denoted two directed graphs. The principle of our method is to find a function of similarity that compute the similarity between each couple of vertices where the fist vertex belongs to graph G_1 and the second vertex belongs to graph G_2. The method extends the principle stipulating that the resemblance of two vertices can be derived from the resemblance of the sets of vertices to which they are connected. It also takes into account the concept of similarity propagation over graph edges. The starting point of our method is a method for information retrieval on the Web proposed by Kleinberg [17] and generalized by Blondel and al [18].

Fig. 10: Hub for data mining domain



When analyzing a graph, the user can design a graph as a model. Examples of such graphs are hub→authority and 1→2→3. ViAGraph computes and shows the similarity of each node of the graph to be analyzed with a selected node from the graph model. The graph model hub→authority is frequently used in information retrieval systems. Many search engines use it to determine the hubs and the authorities among documents to improve the performance of web pages retrieval task. Figure 10 shows a citation graph compared to hub→authority. The main interest of our approach is that it allows the user to define its own analysis methods. Providing the fact that he has a graph model where he associates every node a single well-defined role, he can assume that a node in the studied graph plays a role analogous to the role in the model graph of the node it is most similar to.

6. VI Conclusion and discussion

In this paper, we have described ViAGraph a tool for graph visualization and analysis. This tool implements several known methods and techniques in graph theory and social network analysis. We also proposed novel techniques for better understanding network structures.

An interesting aspect of ViAGraph is its ability to handle multipartite labeled graphs, which comprise nodes, and relationships of several types. This is done by offering visual means to distinguish between the different types of vertices and edges and a personalized data structure to facilitate treatments and to allow constructing and safeguarding graph representation.

A novel method for graph analysis using a graph comparison approach is presented. This method overextends the hubs and authority analysis as used in citation graphs. Work is presently undertaken to adapt this method for graph self-comparison purposes. In other words, we intend to use a graph comparison approach to define a similarity function over the nodes set of a graph.

References

1. H. Purchase, R. Cohen, M. James, *Validating graph drawing aesthetics*, Proc. Graph Drawing'95, LNCS 1027 (1996), 435-446.

2. T. Dkaki, Q.D. Truong, P.J. Charrel, *Visualisation interactive et comparaison de graphes pour l'analyse des réseaux*, in Les cahiers de l'INRIA 2006' (to appear)

3. R. Hanneman, M. Riddle, *Introduction to social network methods*, CA: University of California, Riverside 2005. http://faculty.ucr.edu/~hanneman/

4. insna. http://www.insna.org/.

5. P. Eades, A *heuristic for graph drawing*, Congressus Numerantium, 42:149–160, 1984.

6. T Kamada, S. Kawai, *An algorithm for drawing general undirected graphs*, Information Processing Letters, 31, (l), 7–15 (1989).

7. S. Karouach, Visualisation *interactives pour la découverte de connaissances - Concepts, méthodes et outils*, PhD thesis, Paul Sabatier University-IRIT, Toulouse, France, July 2003.

8. G. K. Stephen, G. Pawel, *Graph drawing with intelligent placement*, In 8th Symposium on Graph Drawing (GD), pages 222–228, 2000.

9. S. Bhowmick D. Dhyani, W. K. Ng, *A survey of web metrics*. ACM Computing Surveys (CSUR), 34(4):469–503, December 2002.

10. M. Delest, G. Melancon, I. Herman, *Indices visuels et métriques combinatoires pour la visualisation de données hiérarchiques*, pages 166–173, Montpellier, 1999.

11. K-P. Yee, D. Fisher, R. Dhamija, Marti Hearst Animated Exploration of Graphs with Radial Layout , InfoVis 2001

12. M. Hall. An r-dimensional quadratic placement algorithm. Management science, pages 219–229, 1970.

13. Lin S. Kernighan B. An efficient heuristic procedure for partitioning of electrical circuits. Bell System Technical Journal, 1970.

14. S. Dongen, *A cluster algorithm for graphs*, Technical report insr0010, National Research Institute for Mathematics and Computer Science in the Netherlands, Amsterdam, May 2000.

15. T. Honkela, *Self-Organizing Maps in Natural Language Processing*, PhD thesis, Helsinki University of Technology, Department of Computer Science and Engineering, Finlad, December 1997.

16. B. Trousse, S. Chelcea, P. Bertrand, *Un nouvel algorithme de classification ascendante 2 - 3 hiérarchique*, In 14th Afrif-afia workshop on Pattern recognition and Artificial Intelligence, Toulouse, 2004.

17. J. M. Kleinberg, *Authoritative sources in a hyperlinked environment*, Journal of the ACM, 46(5):604–632, 1999.

18. M. Heymans, P. Senellart, P. Van Dooren, V. D. Blondel, A. Gajardo, *A measure of similarity between graph vertices: Applications to synonym extraction and web searching*, SIAM Review, 46(4):647–666, 2004.