# On Visualization Techniques Comparison for Large Social Networks Overview: a User Experiment

Bruno Pinaud, Jason Vallet, Guy Melançon

# On Visualization Techniques Comparison for Large Social Networks Overview: a User Experiment

Bruno Pinaud*, Jason Vallet, Guy Melançon

*University of Bordeaux, CNRS UMR 5800 LaBRI, France*

**Abstract**

Visualizing social networks, especially an overview emphasizing their structure, *i.e.*, communities and their interconnections, is known to be a challenging problem. In this paper, we present a set of design rationales to build such overview visualizations of social networks and our solution called JASPER. We evaluate its performances against two of the most wide-spread visualization techniques (matrix and node-link diagram) in a human-computer controlled experiment based on community-related tasks. While none of the techniques emerge as the overwhelming winner, JASPER appears to be one of the best method for each task; a fact sustained by the marks given by the users. Overall, JASPER can be seen as an all-encompassing solution for quickly producing legible and compact overviews of large social networks on a single modern computer.

*Keywords:* Overview Visualization, User Evaluation, Graphs and networks, Social Networks

## 1. Introduction

Along with the constant development of social networks and the continuous raise of their size, social networks analysis (SNA) has been a hot topic for many years [1]. From the vast set of SNA problems, visualization is known to be an efficient [2] albeit challenging analysis technique due to the huge size of social networks. In this paper, we focus on the visualization of the community structure of a network. We propose and evaluate an easy to implement and fast algorithm to do so. When nodes are assembled in communities, one major challenge is to be

*Corresponding author

*Email addresses:* bruno.pinaud@u-bordeaux.fr (Bruno Pinaud), jason.vallet@u-bordeaux.fr (Jason Vallet), guy.melancon@u-bordeaux.fr (Guy Melançon)

able to study how information is jumping in the network between each group [2]. To the best of our knowledge, we are not aware of other algorithm able from an end-user point of view to run on a single workstation and quickly produce a visualization of the community structure while keeping all nodes and edges visible at the same time. Moreover, this kind of visualization is a challenge to obtain as social networks extracted from real-world interactions force us to face several hundreds of thousands [3], when not millions [4, 5], of nodes at once. This easily pushes the existing algorithms [6, 7] built to lay out more modest graphs to their limits.

Thus, we propose JASPER, which stands for *Just A new Space-filling Pixel-oriented layout for large graph ovERview* [8]. Our initial motivation was to create quick overviews of large graphs to allow users to easily get a basic mental map of the targeted dataset before deciding to perform any further visual analytics. Furthermore, because it presents pixel-oriented characteristics and separates nodes in clusters (a.k.a. communities), we found that JASPER could be rather useful to overview propagation and dissemination phenomena in social networks [9]. Nowadays, as everybody is connected and able to communicate with a lot of persons almost instantly by using online social networking services, understanding how information comes to pass using word-of-mouth so quickly is essential. Figure 1 shows a small-multiples visualization of 9 steps of a popular propagation model we have implemented [9]. The growing black area of each thumbnail shows the information spreading. By emphasizing the visualization of communities, JASPER makes the visualization of the cascading effect obvious. As one can see, the propagation hops from community to community. As far as we know, related work about the visualization of propagation phenomenon using a graph visualization is limited. Statistical approaches are often preferred [10] or authors simply use standard node-link diagrams, even when the edge density becomes quite large, thus resulting in very cluttered visualizations [11]. We propose to display graphs quickly using a pixel-oriented representation. When facing large networks, representing every single detail promptly becomes impossible; thus, our solution solely focus on displaying nodes while keeping edges hidden. Consequently, the layout has to be adapted to translate the main information supported by edges, i.e., adjacency. As a result, two nodes close to each other in the final layout would very likely be neighbors (and conversely). Conjointly, we use a space-filling curve to efficiently display every node without any overlapping, as well as to make the most of the available displaying area.

Overall, JASPER can be used to compute quickly snapshots of large social networks in a fixed configuration to study the evolving transmission of information between users. According to the Information Visualization Seeking Mantra (Overview first, zoom and filter, then details-on-demand) [12], JASPER can be used as a quick and simple method to produce a compact overview. Following the same idea with

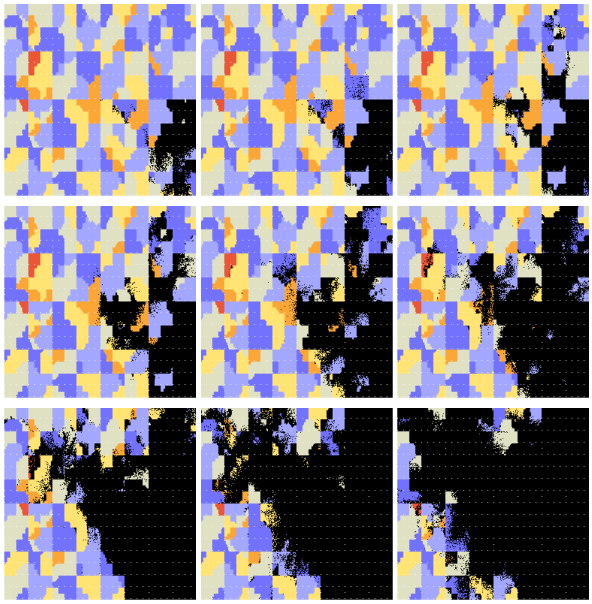


Figure 1: Small-multiples visualization of a propagation phenomenon on a medium size network (65k nodes, 130k edges, 100 communities). Each thumbnail is laid out with JASPER. The growing black area of each thumbnail shows the spreading of the information. Colors are used to display the communities of the network.

the Visual Analytics Mantra (Analyze First - Show the Important - Zoom, Filter and Analyze Further - Details on Demand) [13], JASPER can be used to quickly and easily analyze (*i.e.*, compute communities) and show important information. While it can be applied on smaller graphs, using JASPER in such case may results in diminishing the quality of the visualization. Indeed, other algorithms capable of computing more expressive layouts (showing both nodes and edges, and detailing communities) become able to tackle graphs of such size in a sensible amount of time. Our solution however cannot be used to display dynamic graphs with evolving topologies. This will nonetheless become relevant in future work as the structure of a social network is expected to develop and transform (new users arrive/leave regularly and relations are created/disappear).

This paper builds on and extends a previous work oriented on the presentation of the algorithm behind JASPER and its time and space complexity [8]. Instead, the contributions of the current paper are: 1) guidelines for designing community-oriented visualizations of social networks; 2) a revised presentation of JASPER which is designed to quickly produce an overview of a social network emphasizing communities and their interconnections.

A controlled user evaluation has been added to assess the performance of JASPER for analyzing the community structure of social networks, and compare it to a matrix visualization and two variations around node-link diagrams. In order to ease future work comparing JASPER, we have voluntarily used general tasks (not specific to any kind of data or domain) for community visual analytics, various and freely available datasets (for reproducibility), widely known and used visualizations allowing for indirect comparison with JASPER, and basic interactions (changing the color) which should be available in all visualisation platforms.

The rest of the paper is structured as follows. In Section 2, we define guidelines for designing efficient overview of community-oriented network visualizations and how these guidelines are implemented at each level of the Munzner nested model for visualization [14]. These guidelines give precious hints and structure for looking at related work which is presented in Section 3. Section 4 recalls the algorithm of JASPER and in Section 5, we present the controlled user experiment to properly evaluate it and check if the resulting visualizations are deemed "better" than the more classical node-link and matrix visualizations. We finish this paper with some concluding remarks.

## 2. Design Rationale

We follow the nested model of Munzner [14] which proposes a nested structure. The problems encountered by the user define the operations to perform, which establish in turn the interactions and visual encoding to use, thus eventually characterizing the solution implementation. To initiate the process, we establish three guidelines, described below, that visualization algorithms must meet from a potential user perspective. These guidelines allow to first identify the observer needs, which are for us a community-based visualization of large social networks. They then give potential directions to follow, for developing related work, and eventually a visualization algorithm, along with the required tasks for its evaluation.

*Guideline **G1**.*

When visualizing interactions between users, *e.g.*, propagation phenomena, the visualized social network must indicate the current state of the propagation. More precisely, observers need to know which persons have already received information, which ones are currently susceptible to spread information and which ones are unaware of any information propagation.

*Guideline **G2**.*

Social network users are very often connected with friends, colleagues or family members, and consequently tend to be part of at least one community. Because

an information spread by somebody is more likely to first register on people close to her/him, whom will later repeat it, observers want to know if, and how closely, two persons are related.

*Guideline **G3**.*

Large graphs are complex to handle and visualize but observers interested in propagation phenomena rarely care about complexity, only about the visualization on their screen. Thus, the network must be available as a whole and shown as such. We cannot only "Show the Important" first as claimed by the Visual Analytics mantra. Furthermore, a drawing of the network should appear quickly and not after several minutes of computation.

After defining the guidelines from a user point of view, Munzner's next level leads us to an abstraction into operations on data types. Quite obviously, our data type is a labeled directed graph where nodes represent users of the social network and labels associated to nodes (mapped onto node's color, see below) represent the state of the node about the information propagated. We consider that all users know if they have received the information, if they are willing to spread it further or if they have never heard of it, thus completing **G1**. To address **G2**, and handling real-world problems, we propose to focus the visualization on the community structure of the network instead of the nodes, and communicate the results obtained to the observers. Finally, according to **G3**, we know that the network cannot be altered and that the computation time should be acceptable from a user point of view.

The third nested level requires that we design both visual encodings and interaction techniques. Using specific visual encodings for communities are a perfect way to differentiate those which have spread information from those who have not yet done so, and from the others which are not aware of any information at all. In our case, using colors seems the most obvious and simple choice for both **G1** and **G2**. For instance, nodes from the same community could be drawn with the same color by default but their color should change once they have received information, and once more as they start spreading it. We follow the good practices [15] and propose to use distinctive colors to identify communities (**G2**) and diverse transparency or gray-scale levels, using the alpha channel for instance, to differentiate the different node states (**G1**). The choice of the other visual encodings, like the type of layout or in which fashion nodes and edges are displayed is addressed in Sect. 4.

Finally to complete Munzner's model (interactions), we follow the *Visual Information Seeking Mantra* [12]. In the case of **G1**, *zoom and pan*, and *focus+context* operations are essential to know the state of the different nodes as well as to get more details if asked for. Once joined with the visual clustering of elements or a

distortion of the layout to bring together nodes belonging to the same community, these interactions would also allow to achieve **G2** easily. Finally, the creation of a visualization proposing an overview of the graph, available at any time, is sufficient to address **G3**. That is under the condition that all computations are achieved in a reasonable amount of time.

In complement to the nested model, Munzner proposes a listing of the most common possible threats the visualization solution may be subjected to. While the low-level task validation, *i.e.*, the implementation, requires benchmarks and tests [8], the middle-level ones ultimately need laboratory and field studies for a thorough validation of the solution *i.e.*, expert and user experiments [16]. This is the aim of this paper.

In the following, we propose to take a closer look at existing visual encodings elaborated for large graphs in conjunction with our guidelines.

## 3. Related work

As we may want to visualize hundred of thousands of elements, we concentrate on solutions able to conform to both **G1** and **G2**, such as multiscale visualizations [17, 18, 19]. Basically, the network is hierarchically decomposed into several smaller groups using nested subgraphs. This technique is particularly adapted for large graphs as one can limit the number of level to visualize, thus only displaying part of the hierarchical structure or a subset of the elements according to the hierarchical branch currently tracked. However, this restriction is, at the same time, an advantage and an inconvenient as the resulting representations can be used to visualize large graphs but still do not succeed in showing them in their entirety. These solutions are consequently unable to resolve **G3**.

Decomposing the network into smaller components raises an interesting point; more precisely, if the hierarchical groups coincide with the communities existing in the network, **G2** is solved. Such node grouping can also be used to simplify the layout computation in considering each cluster as a whole entity instead of trying to find an appropriate position for each node separately. This method has been applied to visualize up to fifteen thousand nodes and forty thousand edges [20, 21]. All nodes are individually visible even though the number of elements displayed is quite big thanks to the use of space-filling layouts as a basis to order the clusters in the resulting layout. Such quality is obviously important when dealing with large graphs but more so in our case as we wish to isolate the color of each displayed node. Other algorithms, for instance [22], have proposed a similar method, by ordering nodes along a space-filling curve. These curves are especially designed to "fill" plane areas by placing nodes at regular distances in a given order and pattern [23]. The solution proposed by Muelder and Ma [22] has good advantages

which address **G3**: its computation is quick, the technique can scale up easily, and the produced layouts are, at least from our point of view, clearer than those obtained with force-directed algorithms when displaying larger graphs as a whole. Additionally, the fact that the ordering operation used to place the nodes along the space-filling curve is based on the community structure also provides us with a good candidate solution for **G2**. Nonetheless, the presence of numerous edges still impairs the node visibility in the densest parts of the graph, a problem known in many visualizations [24]. A somewhat similar solution has also been proposed by Auber *et al.* [25] only using hierarchical data, and consequently making it inappropriate for general network visualization.

Using space-filling curves allows to efficiently use space and solve the previous issue. This can however be pushed even further as demonstrated with pixel-oriented visualizations [26]. This technique is commonly used to analyze records and visualize the possible correlations between two measures, one being displayed using the position of the elements along a space-filling curve while the second is depicted using a simple color mapping. Whenever the two analyzed measures are correlated, peculiar color patterns, e.g., chunks of the same color, tend to appear in the representation. Using this kind of visualization would give us the capacity to effectively display all the nodes at the same time (achieving **G1** and **G3**). However, edges –carrying information on the connectedness of each node in the network– cannot be explicitly represented using a pixel-oriented method. Duarte *et al.* [27] have encountered a similar problem. Their *Nmap* layout is introduced as being a neighborhood preservation space-filling algorithm able to display connected nodes close to each other. The resulting visualization is rather reminiscent of Tree-Maps [28, 29] which have been successfully used to visualize very large datasets containing up to a million of elements [30] and to perform social network analysis [31, 32]. By achieving this spatial proximity in the representation to indicate the connection of elements from a structural point of view, as expected for **G2**, edges do not need to be drawn as their existence is hinted by the nodes adjacency in the visualization.

Alternatively, LaGO [33] or Cornac [34] implement visualization and interactive exploration of graphs with millions of elements by using nodes and edges aggregation based on different levels of detail. However, this adaptive level of details is incompatible with our guideline **G1** as information on elements is only accessible after focusing and zooming on certain areas of the visualisation.

Traditional and well-known force-directed approaches, and their parallel or distributed implementations, still receive great attention [35, 36, 37, 38]. These algorithms quickly produce much detailed drawings compared to JASPER. However, these algorithms may require an access to distributed computing platforms and may still need a few minutes to compute, thus failing to help us address **G3**.

## 4. JASPER algorithm for community-based visualization

As mentioned above, considering the huge number of elements we wish to visualize at the same time, we decide to visualize the information in its simplest form, where a node is a pixel. This allows us to represent a lot of elements and attain the overview mentioned in **G3**. Additionally, by changing the pixel's color, one can visualize the state of the elements being represented, thus achieving **G1**. For solving **G2** at the same time, we introduce a visual metaphor to suggest the presence of existing connections between nodes by using a force directed layout algorithm. By laying out the nodes belonging to the same community close to each other, we can suggest that any two nodes next to one another in the visualization are more likely to be connected than any others.

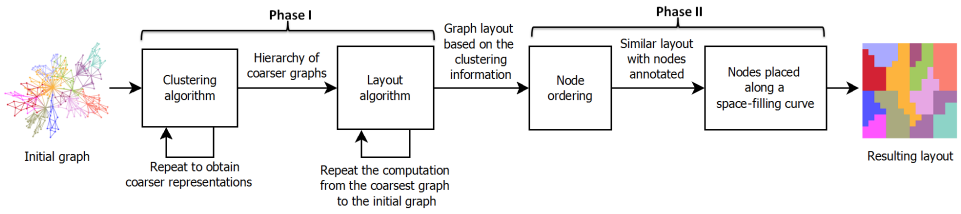



Figure 2: JASPER: from a node-link visualization to a pixel-oriented layout for communities overviewing.

Figure 2 depicts the workflow of JASPER which is composed of two main phases. Phase I computes a layout based on a coarser representation of the initial network. Phase II reorganizes all the nodes of the coarser representation along a space-filling curve accordingly to the spatial position of the nodes obtained in Phase I. The final layout is a space-filling representation with similitude to pixel-oriented layout (only nodes are displayed). We end up with a compact visualization where nodes belonging to the same community are set to be displayed spatially close to each other. A complete illustration of the whole procedure is showed in Figure 3 with a toy example network. Figure 6 shows the results of JASPER along with more traditional visualizations, e.g., node-link diagrams and adjacency matrix. The comparison between all these visualizations is the main research question of the evaluation presented in the next section.

In the first paper on JASPER [8], we present a study of the complexity, computation time and the scalability of the algorithm for graphs with hundreds of thousands of elements to several millions. In this paragraph, we give an outline of some of these results. We achieve computation in about one second for our smallest graph (37k nodes, 370k edges) to about four minutes for our biggest graph (4M nodes, 35M edges) on a mid-2015 workstation laptop. For this latter graph, the longest

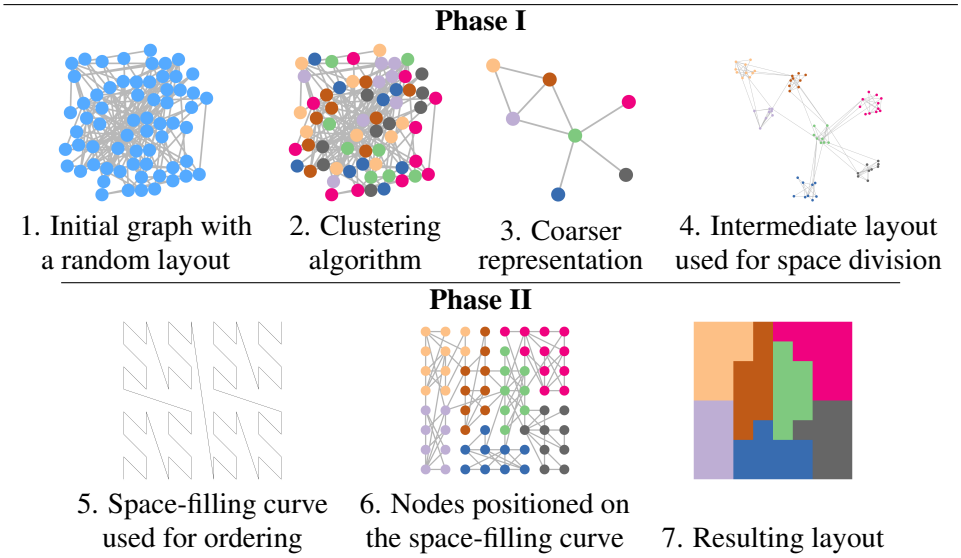



Figure 3: Illustration of Phase I and II on a toy network.

operation is the clustering algorithm. On average, we end up being able to compute the layout on relatively large graphs (2M nodes and 5M edges) in approximately 35 seconds. Note that the space-filling and node ordering computations are based on a multi-threaded implementation. Because the number of edges is always bigger than the number of nodes in a real-world network, sometimes up to an order of magnitude, we can predict that the number of edges, and more precisely, the part of the solution whose complexity is impacted by it, *i.e.*, the clustering algorithm, will be our pitfall in terms of efficiency. Overall the performances of Jasper are closely correlated to the performance of the clustering algorithm. However, our solution has the advantage of being highly modular as the clustering algorithm detecting communities and the layout algorithm drawing the coarse graph representations can be changed at will should alternative algorithms offering better results appear.

### *4.1. Phase I: Build and Lay Out a Coarser Graph*

As we are tackling large graphs, it is known that many layout algorithms generally need a long computation time to produce a quality layout. The quality of a drawing is often improved by reducing edge crossings, or minimizing edge length [39]. Thus, it is directly linked to the number of elements (nodes and edges) of the graph. If we are able to keep only the most important nodes and edges, we should be able to compute a quality layout much more quickly and efficiently. This reduced graph is often called a coarser or skeleton graph. Similar approaches are

used in different layout algorithms and analysis techniques [18, 40, 41, 21, 42, 43]. Overall due to its cluster-based nature, using Jasper to visualize a complete graph (counting only one cluster) does not make much sense. On the other hand, a lot of very small clusters (consisting only of very few nodes) would not greatly improve the execution time of the layout algorithm. Thus a right balance must be found by the clustering algorithm.

#### *4.1.1. Clustering nodes*

Jasper is more an abstract framework than an exact step-by-step list of operations. So, any clustering algorithm can be used. According to Fortunato [44], and with the additional support expressed by Didimo and Montecchiani [21], we decide to use the Louvain algorithm [45] thanks to its performance. Evaluating the difference with other clustering methods is left for future work.

As shown in Figure 3-1, nodes are considered homogeneous in the initial graph (all nodes have the same color). Once we apply the clustering algorithm, nodes of the same cluster are colored similarly to identify them (Figure 3-2). We entirely rely on the clustering algorithm efficiency to appropriately regroup each node with its proper and closest relatives.

#### *4.1.2. Building on clusters*

Once each node is set in the adequate cluster, we create a map of their relations to obtain a first coarse graph (Figure 3-3). Repeating the clustering step several times will produce coarser and coarser representations. More generally, upon each computation of a coarser graph, we first group close nodes together then communities linked to one another. We use the groupings herein created to decide how close two nodes should be displayed.

Using a more formal approach, coarser representations can be described as follows. For a graph $G_k = (N_k, E_k)$ identified by a set of nodes $N_k$ and a set of edges $E_k$, where $k \geq 0$; we define $G_0$ as the initial graph we want to visualize and $G_1, G_2, \ldots, G_p$ as its gradually coarser representations. By applying a clustering algorithm on $G_i$, where $i \geq 0$, a set of clusters $C_i$ is created such as each node $n \in N_i$ belongs to one and only one cluster $c \in C_i$ (described as $cluster(n) = c$). Those groupings are then translated to a coarser representation of the graph, *i.e.*, for each cluster $c$ in $G_i$, a node $m$ is created in a new graph $G_{i+1}$ such as $m \in N_{i+1}$ and, if an edge $e \in E_i$ exists between two nodes $n$ and $n' \in N_i$ in distinct clusters ($cluster(n) \neq cluster(n')$), then, an edge $\epsilon \in E_{i+1}$ is set between the two nodes $m$ and $m' \in N_{i+1}$ –respectively representing $cluster(n)$ and $cluster(n')$. This edge linking $m$ and $m'$, can also have a weight based on the total number of connections between the nodes contained in each cluster represented by $m$ and $m'$ or the aggregation of any existing weight. This weight may be used to improve the layout computed at the next step.

*4.1.3. Layout computation*

After laying out the coarsest graph $G_p$, we gradually use the layout computed on the clusters in $G_i$ (where $p \geq i > 0$) to place the subgroups in $G_{i-1}$. The operation is repeated until $G_0$ is reached and treated (Figure 3-4).

To perform layout computations, we have to choose an efficient layout algorithm offering legible drawings in a small amount of time. We chose $FM^3$ ("Fast Multipole Multilevel Method") [46] which proposes a good balance between execution time and drawing quality with minimal edge-crossings and overlapping edges [47, 48, 49]. Those two points are essentials as we wish to keep a readable layout of the clusters to easily distinguish one from another.

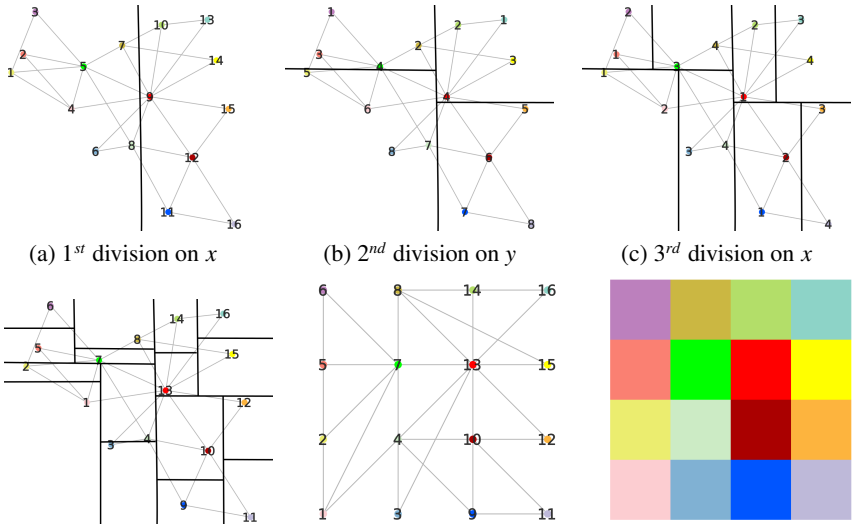
Computing the layout of the coarsest graph $G_p$ is straightforward as it simply requires to apply the algorithm layout on $G_p$. The layout of any of the following coarser graphs $G_i$, where $i < p$, needs a few additional steps. First, a local layout is computed in each existing cluster $c \in C_i$ by only considering the corresponding subgraphs. Then, freshly computed coordinates of each node $n \in N_i$ attached to the cluster $c$ (with $cluster(n) = c$) are transformed through translation and scaling so that all nodes in $c$ are displayed in a (small) area as defined by $m \in N_{i+1}$, representing $c$ in the coarser graph $G_{i+1}$. Each node coordinate in a coarser graph $G_{i+1}$ is thus used as an anchoring point to lay out the nodes of $G_i$. These two steps are repeated for decreasing values of $i$ until we reach $G_0$ and have computed the coordinates of each initial node (Figure 3-4).

On a more general note concerning Phase I, the rapidly decreasing number of nodes for each coarser representation makes the construction of $G_{i+1}$ and each application of the clustering and layout algorithms on it quicker. This is not surprising considering the graphs obtained from the clusters are getting simpler, however, based on observations, repeated applications to obtain several levels of coarse graphs do not necessarily improve the resulting layouts. As a result, in our applications, one coarser representation ($G_1$) has proved to be sufficient more often than not, with a second level of coarse representation only appearing to be of interest when used on some of the largest graphs. We offer nonetheless the general method as such, considering structured graphs with established hierarchical communities will benefit from it. In any case, the appropriate number of coarse levels is left to be decided upon application.

*4.2. Phase II: Order Nodes to Produce a Pixel-Oriented layout*

Pixel-oriented visualizations have been presented by Keim [26] to display important quantities of data in a minimal space. Elements to display are ordered and placed along a space-filling curve. After ordering nodes according to their spatial

position by the end of Phase I, we first divide the space using a technique similar to the one used to create $k$-d trees [50]. When each node is isolated from its neighbors, they are ordered according to a space-filling pattern and are laid on the corresponding curve at their given position. Finally, the size and shape of nodes are maximized and edges are finally hidden to obtain the maximum legibility and avoid overlapping elements (Figures 3-4 to 3-7). As shown in Figure 4, the successive application of these three steps allows us to create a compact representation where nodes connected to each other end up in the same vicinity.



(a) $1^{st}$ division on $x$ (b) $2^{nd}$ division on $y$ (c) $3^{rd}$ division on $x$

(d) Once isolated, nodes are ordered along the space-filling curve (e) Nodes are positioned according to the curve (f) Node's sizes and shapes are changed

Figure 4: Illustration of Phase II on a toy graph (16 nodes) to produce a pixel-oriented layout. For each division (only the first 3 steps here, (a) to (c)) nodes are ordered according to the dimension considered (alternating between divisions) into two subsets of equivalent size. Divisions are done until each node is isolated (d). The final ordering (d) and placement along the space-filling curve (e) is done following a N-order space-filling curve layout.

### *4.2.1. Space division and node's ordering*

The node's ordering is done in a way similar to the one employed for a $k$-d tree construction. A $k$-d tree is a data structure used for the storage of $k$-dimensional data. In our case, the dimensions considered are those specified by the layout ($x$ and $y$ coordinates). We first divide the graph in two parts with a similar number of nodes in each halves using a pivot value on the first dimension (Fig. 4a). The two resulting halves are then divided again in two equal parts but using a pivot value on

the second dimension (Fig. 4b). This process is repeated (Fig. 4c) until each node is isolated and thus is given an order (Figure 4d).

*4.2.2. Placement along a space-filling curve*

The choice of the space-filling curve is important as it indicates how the elements are to be divided and ultimately ordered. We use the space-filling curve popularized by Morton [23] using an *N* shape as shown in Figure 3-5. Similar results can be obtained with different curve designs or orientations –such as the Hilbert or the Z-order curves– as long as the node ordering during the division respects the path followed by the curve. This last point is crucial to preserve adjacency in the final layout.

Space divisions along *x* and *y* axis give an appropriate order for each node to be displayed along our space-filling curve as we establish a correspondence between each tile and the points on the Morton curve (Fig. 4e). Once ordered (Fig. 3-6), nodes can be laid out according to the curve shape.

*4.2.3. Coloring, Reshaping and Resizing nodes*

Finally, nodes are reshaped and resized to obtain a more evenly tilled depiction (Fig. 3-7) akin to a pixel oriented representation (see Fig. 4f). Each node's color is also changed according to the clustering algorithm, thus coloring similarly nodes belonging to the same communities. It is important to note here that while some heuristics and methodologies are well-known to help in picking appropriate colors for visualization purposes [51, 52] finding color scales adapted for proposing hundreds of easily differentiating hues is not possible.

*4.2.4. Discussion*

As we know the size of the graph to lay out, we can compute from the start how many divisions will be necessary to isolate all nodes. However, if we wish to conserve a regular shape for the final layout, the number of divisions have to be equal for each halves. This means regions with fewer nodes have to incorporate "holes", that is white-spaces which are given an order too. Moreover, because the number of nodes is equally distributed in each halves during the division, the "holes" are evenly spread throughout the whole space-filling representation. The appearance of these white spaces is noticeable on representations with a limited number of nodes, but become less visible when facing large graphs. While the ideal visualization ratio is set to be 1:1 (square shape), the successive divisions and the rearrangement along the space-filling curve do not give us the possibility to freely rearrange nodes. However, to avoid creating representations with too many white spaces, we suggest to use a variable shape for the representation depending of the number of nodes to display. The overall resulting Morton curve is composed of

$n$ N-shaped patterns (Fig. 3-5) and the layout is either a square (total of $4^n$ possible points) or a rectangle (number of points of $2 \times 4^n$ points for a complete rectangle). A variable layout shape allows us to avoid incorporating too many "holes" and to use most of the space available by following a simple rule: let $n \in \mathbb{N}$, a natural number, and $|V|$ the number of nodes in the graph to display. Generally speaking, the resulting shape will be a square if $2 \times 4^{n-1} < |V| \leq 4^n$ and a rectangle otherwise (when $4^n < |V| \leq 2 \times 4^n$). We test the number of nodes during the first space division to either spread the data on a square or on half that surface, i.e., a rectangle. This is implemented by either using the whole space-filling curve or only half of it, if the number of nodes allows it.

## 5. User evaluation of Jasper

Following Purchase methodology [16], we designed a within-participants experimentation to evaluate Jasper as a compact social network overview visualization algorithm against well-known and widely used methods. The experimentation is motivated by the following research question:

*Is* Jasper *a better solution for analyzing the community structure of social network compared to two variations of node-link diagrams and a matrix visualization?*

Although Jasper was originally designed for generating overviews of large graphs, we expect it to allow users to successfully and quickly perform some basic community oriented tasks. As a consequence, we evaluate more general tasks involved in the study and identification of communities in social networks.

Overall the experiment is composed of 4 visual conditions used with 4 datasets and 3 tasks to be able to generate $4 \times 4 \times 3 = 48$ questions per user.

### *5.1. Datasets*

We use 4 real-world datasets of different sizes plus another small random network generated following the Wang *et al.* model [53] with 10k nodes and twice as much edges for user training purposes. The different sizes of the datasets are used to analyze the responses of users depending of the layout used, but also on both the size and density of the networks being visualized. All the following datasets are freely available as part of the Stanford Large Network Dataset Collection [54].

*D1 – Enron emails*

The network is composed of 33,696 nodes and 180,811 edges and represents exchanges of email messages between employees of the Enron corporation [55].

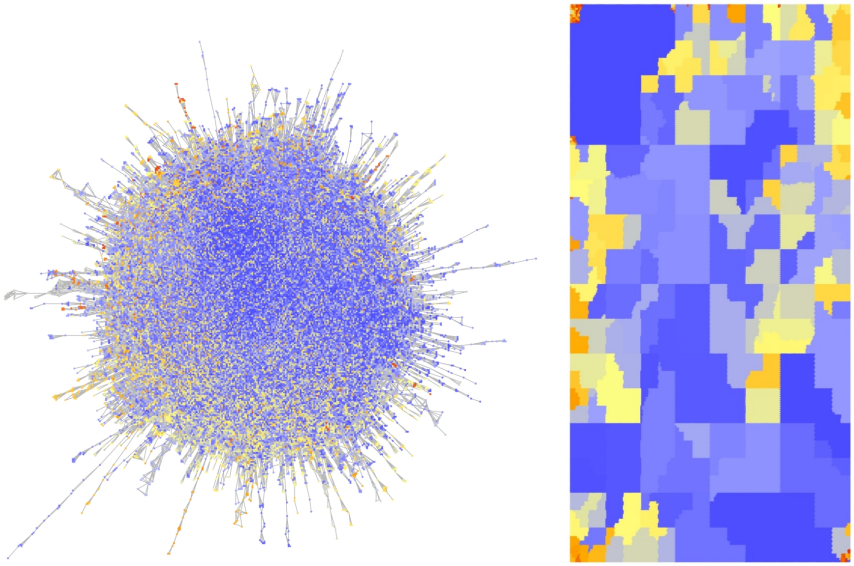
*D2 – Slashdot*

The network is composed of 82,168 nodes and 948,464 edges. It is a snapshot from February 2009 of the user community from the Slashdot website (`https://slashdot.org/`). Although in the initial dataset, users have tagged each other positively or negatively, we are solely interested by the topological network created by those interactions.

*D3 – Epinions*

The network is composed of 119,130 nodes and 833,695 edges extracted from the (now closed) Epinions website [4]. More specifically, as a review website, Epinions was proposing users to select other users and "trust" them as well as their opinions on a wide range of products sorted by categories.

*D4 – DBLP*



(a) Node-link layout with $FM^3$ (b) Final result with JASPER

Figure 5: Application of JASPER on the DBLP graph [3] (317,080 nodes, 1,049,866 edges and 430 clusters).

This is our largest network with 317,080 nodes, 1,049,866 edges and 430 clusters (Figure 5). It is a filtered version of the co-authorship network extracted from the DBLP bibliography website (`http://dblp.uni-trier.de/`).

To visualize networks with at least a common ground, **D1** and **D2**, which

are composed of several disconnected groups, have been filtered to only keep the largest connected component. Although we designed JASPER as a solution to represent the overview of rather large graphs, one cannot help but notice that the size of the graphs we have selected for our evaluation is more modest than what we have presented in our initial work on JASPER [8]. While JASPER is able to handle very large graphs, we deem it unfair to evaluate the other visual conditions on such extreme criterion.

### *5.2. Tasks*

Considering our strong interest towards social networks, an important point of interest concerns the study of communities and the social ties existing between them. We thus propose solely community-oriented tasks, allowing us to measure how efficiently the different visualizations can represent this information, even for an overview of the network.

*T1 – Which highlighted community is the largest?*

Obviously, and beyond our guidelines, communities have to be distinguishable from one another along with their relative sizes. As a community delimits a group of people with strong and numerous ties with one another, it very often possesses a high edge/node ratio. Large communities are remarkable because this density is spread between many individuals. When analyzed, one will often want to identify these sizable groups, isolate them, and study them aside.

*T2 – How many highlighted communities is there?*

Although highlighting is an efficient way to bring information into focus (*i.e.*, for addressing Guidelines **G1** and **G2**), one still needs to be able to discern this visual feedback. For this task, we extend this requirement by displaying several highlighted communities at the same time in a single graph. This allows us to measure how efficiently users can identify precise communities and differentiate them from the others.

*T3 – Are the highlighted communities connected?*

In the same way that most usual graph visualizations try to legibly indicate the existence of a link between two nodes, we wish to know if two communities are actually connected, *i.e.*, if there is at least one node from the first community connected to one node of the second community. We propose this task to evaluate the adjacency-rendering abilities of the different visualizations. This task also address Guidelines **G1** and **G2**.

### *5.3. Visual Conditions*

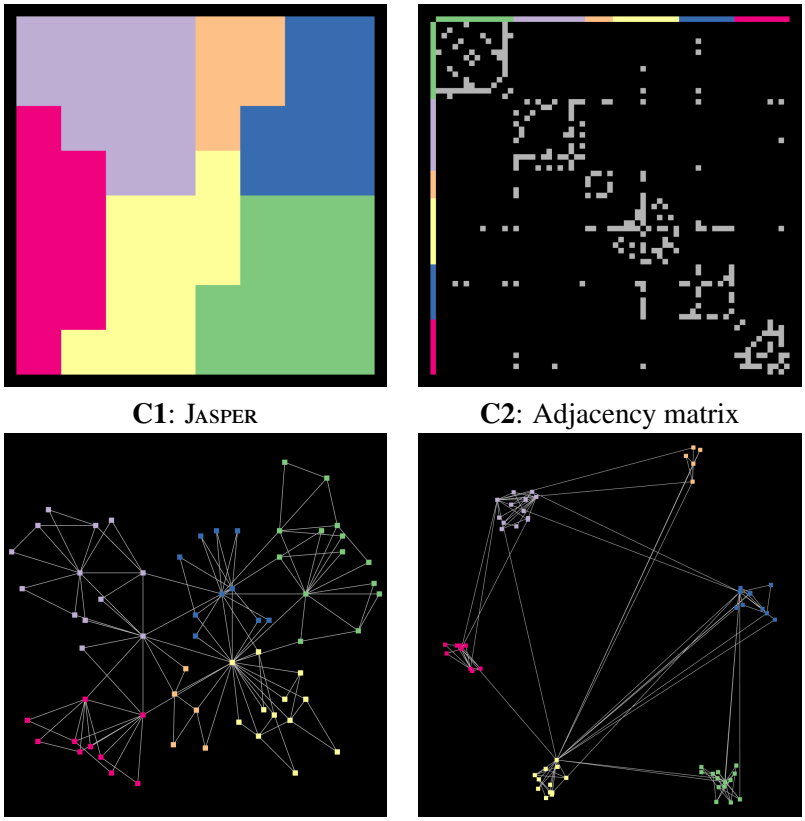
We choose to compare JASPER with three other well-known visualizations able to produce drawings quickly to address Guideline **G3**: an adjacency matrix and two variations around a node-link diagram (Figure 6). In order to be fair throughout our experiment with respect to the nature of the tasks being community-oriented, we propose two different layouts to draw the graphs with the node-link diagram. Although both use a force-directed layout, one additionally propose a more noticeable spatial separation of the communities. For each graph, we use the Louvain clustering algorithm to identify the communities and mark accordingly each node with an index specifying its cluster number. Based on this index, we then colored the nodes using a red-yellow-blue color scale (qualified as colorblind safe on ColorBrewer [52]), so that each community color is different. This coloring is given for each graph and is kept across all conditions. Furthermore, for all visualizations, nodes are drawn using square shapes, edges are colored in gray and drawn as straight lines on a black background.

*C1* – JASPER

As introduced above, JASPER is an adjacency-preserving, space-filling, and pixel-oriented layout. As a preliminary hypothesis, we believe our solution to be particularly adapted for an overview, and thus, should be the most efficient for finding communities and evaluating their size (**T1**). It should also perform quite well with any community numbering related task (**T2**), due to its design. Its space-filling characteristic however is likely to impede to a certain extent the acknowledgment of existing connections between communities, thus likely giving poor results for task **T3**.

*C2 – Adjacency matrix diagram*

This is the visualization of an adjacency matrix. As the main characteristic responsible for making a difference between two adjacency matrix diagrams is the selected node ordering, we arrange the nodes using the cluster indices. As a preliminary hypothesis, we believe the adjacency matrix diagram to be a rather dense and complex representation lacking the appeal offered by other visualization techniques. Nonetheless, the matrix still shows all the information needed to properly complete each task. Its only weakness is the difficulty for users unfamiliar with this representation to gather this knowledge [56]. We thus expect an overall fair rate of accurate answers but with probably longer response times than the other conditions for all the tasks.



**C1**: JASPER

**C2**: Adjacency matrix

**C3**: Node-link diagram (NLD)

**C4**: NLD with community separation (NLD-com)

Figure 6: The four different conditions used during the experiment with the same graph. Nodes are shaped and colored identically across all conditions.

*C3 – Node-link diagram (NLD)*

We use this condition with the well-known force-directed layout $FM^3$ [46]. While certainly the most common and preferred visualization [57], NLDs are known to provide poor visualization for dense graphs with only a few hundred nodes when considering classical graph visualization tasks (e.g., counting nodes, finding paths [24]). As the tasks considered for our experiment are community-oriented, we may possibly end up with very different results. However, with the visualization being unsatisfactorily, especially for large graphs, we state as a preliminary hypothesis that NLD remains unlikely to excel at any of the given tasks.

*C4 – NLD with community separation (NLD-com)*

This condition uses a layout similar to the intermediate layout obtained based the coarser graph of JASPER at the end of Phase I. Compared to the standard NLD (**C3**), it shows the communities with a much more legible view of the graph. It is computed using $FM^3$ and the communities are displayed more densely, allowing nodes belonging to the same community to be spatially closer to their neighbors than outsider nodes. The preliminary hypothesis formulated when using this layout is more positive than the previous one when considering **T2** and **T3**. Indeed, placing nodes of a community close together improves the visibility and the identification of the communities. When laid out this way, the connections between groups of users are also quite discernible from connections between users of the same community. On the other hand, task **T1** remains precarious as nodes from the same community can appear more and more indistinguishable from one another especially for large networks.

### *5.4. Evaluation setup*

All users used the same computer (no updates during the experimentation), same screen (same resolution and brightness for each user), same room with the same light conditions, same mouse, same software (no recompilation). We use two different series of questions for each dataset, task, and visualization combination. The layouts are similar but the communities highlighted (i.e., the subjects of the question) are different. This allows us to further validate the solution by using an alternative scenario to overcome a learning effect.

The communities used for each task were chosen among the largest communities built by the Louvain algorithm. For **T1**, the large community has about 10% more nodes than the small one. For **T2**, the comparison was done between 3 and 4 communities. For **T3**, the available choices were limited by the datasets but we always chose communities connected with numerous connections between them. To perform the experimentation, we use a simple web application, proposing for each question two pictures of the same graph with the same layout under the same condition but with different highlighted communities in pink. Edges on the other hand remain of the same color (gray). The evaluation was done in French which is the native language of all participants. We used TULIP [58] to produce the pictures. Fig. 7 shows a screenshot of the web user interface. Users have to choose the drawing with the largest highlighted community (**T1**), the higher number of highlighted communities (**T2**) and the smallest distance between the highlighted communities (**T3**). We also offer a "Je ne sais pas" (I do not know) button to go to the next question without waiting for the remaining time.

The task to complete is expressed at the top, with more detailed instructions given underneath, and a time counter placed at the top-right corner to indicate the

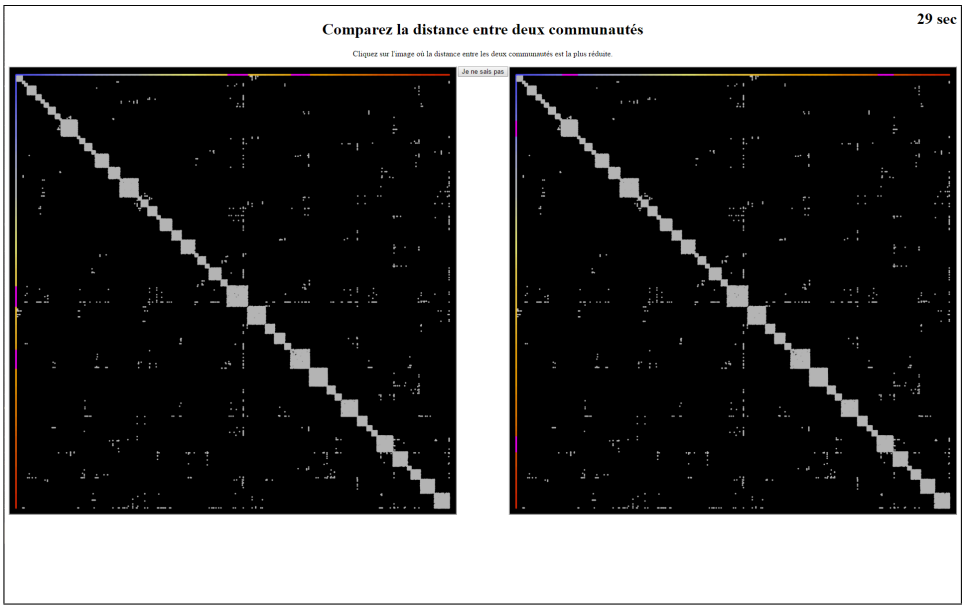



Figure 7: Graphical user interface used for the evaluation. The task is specified at the top of the page; the remaining time is at the top-right corner; the two pictures of the current graph are displayed on the left and right sides; the "Je ne sais pas" ("I do not know") button is located between the two visual conditions. Users have to click on the image they choose.

remaining time to answer (maximum of 45 sec.). Past that time, users can still look at the visualization for as long as they need but any given answer will be marked as false. We hope that by leaving them the time to study a graph visualization in depth, we give them the opportunity to forfeit one question for a better understanding of the visualization, thus leading to better result with the remaining tasks. While the pictures are static drawings, an interaction is offered to the users: the possibility to hide the highlighted communities whenever the picture is hovered by the mouse cursor. Moreover, the color of the communities remains similar across all conditions, only changing when the community is highlighted. All these parameters, especially the interaction which was not initially planned, were decided and verified during 3 pilot experiments with different users unfamiliar with our work.

Overall, the experiment is divided into 4 sets of questions, each set using only a single condition. The ordering of tasks, graphs, and conditions is managed with a Latin square to avoid identical scenarios to take place between users. To keep users attentive all along the experiment, short intermissions of up to two minutes are proposed between each set of questions.

When starting the evaluation, users are not familiar with the different conditions or tasks. For users to perform at their utmost capability, we use a total of six

training questions per question set. The first three are used to introduce the tasks with the current condition on the training dataset, and give users the opportunity to ask for help or additional explanations concerning the visualization. The remaining three training questions are simply standard questions, selected at random in the alternative series whose results are ignored. Each user is thus asked to answer to 6 training questions $\times$ 4 conditions + 48 normal questions = 72 questions.

After completing the experiment, users have to answer a survey asking them to sort the conditions for each given task according to their preferences, feelings of efficiency, and overall to choose their favorite condition. An inquiry for diverse remarks is also performed.

### *5.5. Results*

For the whole experimentation (excluding pilot experiments), we had a total of 26 participants formed of associate professors, researchers, engineers, post-doctorates and PhD students all with a computer-science background. They were asked to have at least a basic understanding of graph theory (undergraduate level) but no prior knowledge concerning information visualization or network analysis.

To analyze the error rate, we use a Pearson's Chi-squared test, allowing us to verify if there is a significant difference between all conditions. Then, following the result, we use a post-hoc pair-wise analysis with a Wilcoxon's rank-sum test to identify where the significant difference really is. For the response time analysis, we first use a Kruskal-Wallis' rank-sum test and again a Wilcoxon's rank-sum test as a post-hoc pair-wise analysis. One can notice that we use non-parametric tests for both the error-rate and the response time, as both sets of results follow non-normal distributions. We use a standard significance level $\alpha = 0.05$ to declare whether a significant difference exists or not. Figure 8 shows the overall results for response times and error rates for each task and each condition.

*T1 – Large community detection.*

Jasper (**C1**) and Matrix (**C2**) give the lowest error rates significantly below than NLD (**C3**) and NLD-com (**C4**). Response times for **C1** and **C2** are also better but the difference is only significant when compared to **C4**.

*T2 – Number of highlighted communities.*

For this task, a huge gap appears between **C3** and the three other conditions in terms of error-rate. **C4** gives here a significantly lower error-rate than Matrix (**C2**) while Jasper (**C1**) sits halfway in-between. The response time analysis shows only small variations, and, while the average NLD response time seems to be longer, no significant difference is found.

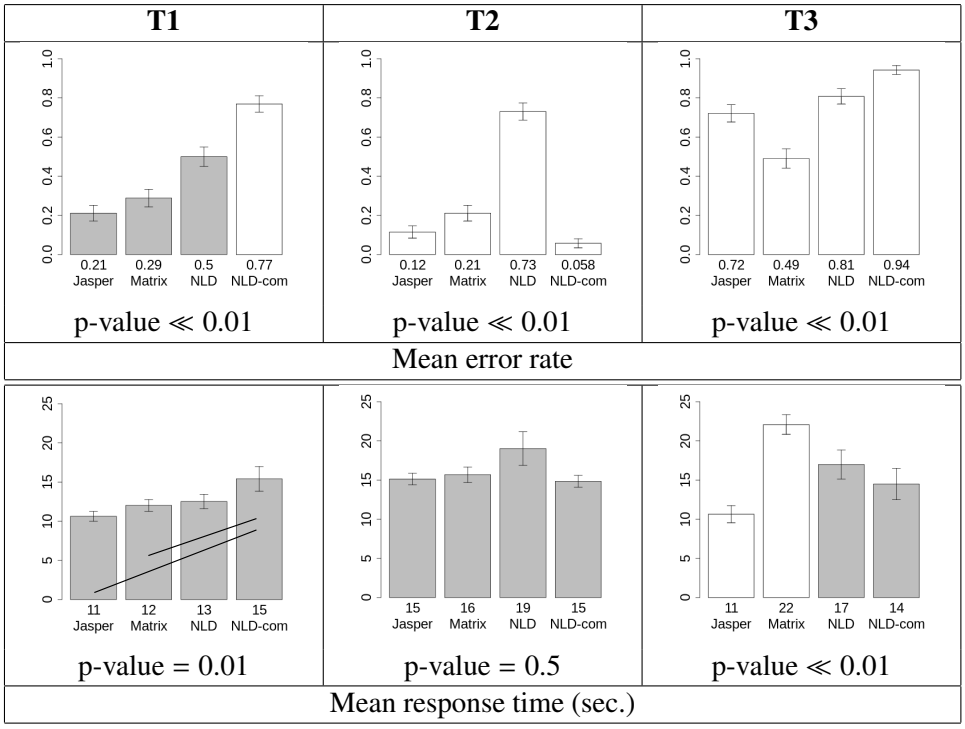



Figure 8: Mean error-rates and response times for each task. For each diagram, the lower the value is, the better. All $p$-values$< 0.05$ indicate significant differences between results. Bold black lines between bars indicate where a significant difference is found. A white bar means there is a significant difference with all other bars. The mean value is written below each bar.

*T3 – Distance between highlighted communities.*

This task proved to be much more complicated than initially anticipated in the pilot experimentation as shown by the high error-rates. Matrix (**C2**) returns the lowest error rate by far, with **C1** and **C3** coming next. However, the average response time when using JASPER is significantly lower than all the other conditions, it even cuts by half the average time needed to answer when using **C2**.

On a side note, while all datasets have been chosen to bring the experimentation as close as possible to a real case, the selected networks are not identical and some statistically significant variations exist for some tasks. For instance, even with a Bonferroni correction ($\alpha = 0.025$), the successful completion of **T1** takes significantly less time on DBLP (**D4**) and the completion of **T3** has been significantly more successful on Epinion (**D3**) –with still 49% of errors– as well as seemingly faster –but not significantly– than on the other networks. Task **T2** on the other

hand does not show any significant differences in the response times and error rates across the different datasets.

*Qualitative comments*

We then analyze the qualitative comments and marks gathered with the survey given after each evaluation. We also use a Kruskal-Wallis' rank-sum test to compare for each task users appreciation concerning the different conditions (Tab. 1). We asked users to rank conditions according to their feelings for each task from 1 (best) to 4 (worst, equality forbidden). This ranking was made without any knowledge of the experiments results. The average ranks are rather conform to the measured response times and errors. For **T1**, JASPER (**C1**) and Matrix (**C2**) are both preferred to node-link visualizations. Task **T2** however brings NLD-com (**C4**), **C1** and **C2** at the same level due to mixed opinions with **C3** being mostly rated low. **C4** and **C1** end up with the best score for **T3**, following the response time tendency but ignoring their high error rates (especially for **C4**).

| Task | **C1.** JASPER | **C2.** Matrix | **C3.** NLD | **C4.** NLD-com |
|---|---|---|---|---|
| **T1** | 1.64 | 1.96 | 3.16 | 3.24 |
| **T2** | 2.12 | 2.20 | 3.64 | 2.04 |
| **T3** | 1.80 | 3.20 | 3.28 | 1.72 |

Table 1: Average score given (the lowest, the better) by users after the experiment for each condition and each task.

### *5.6. Discussion*

Overall, while no conditions emerge as the clear winner, we have some interesting trends for each task. First, with **T1**, the ranking obtained on the error-rate confirms the ability of JASPER to successfully solve this task, as initially expected in Section 5.3. There is, however, no significant difference of response time between JASPER (**C1**), Matrix (**C2**) and NLD (**C3**), although the error rate is quite high (more than 20%). Task **T2** sees the prevalence on all fronts of NLD-com (**C4**) with JASPER not so far away. This is because nodes are arranged as tight clusters and can be easily identified, even for the largest networks. When compared to the standard NLD (**C3**), one can clearly see the interest in regrouping nodes as communities, nevertheless, this advantage turns to be a double-edged sword as **C4** falls back on the two other tasks whereas NLD (**C3**) presents average results instead. Finally, **T3** has certainly the most striking results with its high error-rate. While, after the pilot experiments, we were aware of the high complexity of this task, we were far from expecting such poor results in general. As the identification of neighboring elements is one of the atomic operations one would expect to perform on a

node, the generalization of this task to a community seemed a necessary operation in a context where communities are essentials. In the end, while the task may have been too intricate, we believe that the proposed conditions are also not adapted; alternatively, a multi-scale visualization may be a much more efficient solution for such task [59].

Overall, and despite its popularity, NLD (**C3**) does not excel anywhere and gives almost consistently worst results than Matrix (**C2**). Its variation NLD-com (**C4**) proposes extremely improved results on **T2** in terms of error-rate but this condition does not bring improvements anywhere else. **C2** showed the expected efficiency but the responses are faster than what we had initially predicted except for **T3**. This proves that, given a good ordering of the nodes, the adjacency matrix diagram is still a relevant visualization even for moderately large graphs. For both **T1** and **T2**, JASPER comes as one of the best with no significant differences found in the experimentation results. Task **T3** shows more mixed results with a slightly better-than-average error-rate, in comparison to the other conditions, but also with the best response time, coming at half the time needed for the Matrix. So while JASPER is not the best at every task, its global performances seem to mark it as an all-encompassing acceptable overview solution, a fact sustained by the marks given by the users during the final survey.

With the results of the user's evaluation at hand, it is necessary to acknowledge the limitations of our method. As we emphasize the overview characteristic of the resulting visualization, we are aware that any analysis or in-depth studies on a graph can not be achieved using our method alone. Obviously, if one tries to discover all there is to know about a graph using a single approach, it will never be sufficient to understand every singular detail; attaining this outcome will indeed require different methods and points of view. Only then, may it ultimately result in a more complete picture. This is true for our method as, due to our focus on node representation, we have hidden edges, thus occluding some information to the observer. Even though we use node placement and colors to give hints of detected communities and existing connections, such metaphors are far from perfect and the visualization produced is not entirely sound. Nevertheless, with the assistance of supplementary interactions and additional visualizations, more insights on the structure of the graph and connections established through edges can be easily accentuated. An interaction like a neighborhood highlighter [60], or a compound visualization like matrix/node-link drawing [61] focusing on specific sub-graphs are such tools. Furthermore, this lack of explicit information displayed is not solely encountered when using our method as large graphs typically contain more information and details most screen can display nowadays. Even other solutions similar to JASPER, using pixel-oriented layout to maximize the amount of information displayed in limited space, can only give an approximation [62].

Multi-level solutions [49, 19, 33] can overcome such restrictions but only up to a certain level as their resulting layouts solely provide an approximated or partial representation of the graph.

## 6. Conclusion

We have presented and evaluated JASPER, a community oriented visualization algorithm for large networks. Its space-filling and pixel-oriented characteristics are well suited to display a fast yet intelligible overview of the networks with node adjacency preserving qualities. The user evaluation proved our solution to be quite efficient on community-related tasks, leading us to believe that JASPER can also be useful in situations where a visualization to assess the state of a large graph at a glance is required. As networks are still getting larger and larger as more data is harvested through data-science techniques, specific visualization methods will have to be developed to cope with the ever increasing number of elements to consider. Although, in the end, none of the conditions considered during the evaluation have proved to be better than the others for community-oriented tasks on large graphs, we believe the issue to be particularly worthy of interest and an important research topic to explore further.

## References


[1] S. Wasserman, K. Faust, Social Network Analysis – Methods and Applications, Cambridge University Press, 1997.

[2] J. Golbeck, Analyzing the Social Web, Morgan Kaufmann, 2013.

[3] J. Leskovec, K. Lang, A. Dasgupta, M. Mahoney, Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters, CoRR abs/0810.1355 (2008).

[4] J. Leskovec, D. Huttenlocher, J. Kleinberg, Signed networks in social media, in: Proc. of the SIGCHI Conf. on Human Factors in Computing Systems, CHI '10, ACM, 2010, pp. 1361–1370. doi:10.1145/1753326.1753532.

[5] J. Leskovec, D. Huttenlocher, J. Kleinberg, Predicting positive and negative links in online social networks, in: Proc. of the $19^{th}$ Int. Conf. on World Wide Web, ACM, 2010, pp. 641–650. doi:10.1145/1772690.1772756.

[6] T. von Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J. J. van Wijk, J.-D. Fekete, D. W. Fellner, Visual analysis of large graphs: State-of-the-art and future research challenges, Computer Graphics Forum 30 (6) (2011) 1719–1749. doi:10.1111/j.1467-8659.2011.01898.x.

[7] F. Beck, M. Burch, S. Diehl, D. Weiskopf, The state of the art in visualizing dynamic graphs, in: EuroVis - STARs, Eurographics Association, 2014, pp. 83–103. doi:10.2312/eurovisstar.20141174.

[8] J. Vallet, G. Melançon, B. Pinaud, JASPER: Just A new Space-filling and Pixel-oriented layout for large graph ovERview, in: Conf. on Visualization and Data Analysis (VDA 2016), Electronic Imaging, 2016, pp. 1–10. doi:10.2352/ISSN.2470-1173.2016.1.VDA-484.

[9] M. Fernandez, H. Kirchner, B. Pinaud, J. Vallet, Labelled Graph Strategic Rewriting for Social Networks, J. of Logical and Algebraic Methods in Programming 96 (C) (2018) 12–40. doi:10.1016/j.jlamp.2017.12.005.

[10] B. Wang, Y. Sun, C. Tang, Y. Liu, A visualization toolkit for online social network propagation and influence analysis with content features, in: 2014 International Conf. on Orange Technologies, 2014, pp. 129–132. doi:10.1109/ICOT.2014.6956616.

[11] J. Lu, X. Yu, W. Wan, Visualization research of the tweet diffusion in the microblog network, in: Int. Conf. on Audio, Language and Image Processing, 2014, pp. 592–595. doi:10.1109/ICALIP.2014.7009863.

[12] B. Shneiderman, The eyes have it: A task by data type taxonomy for information visualizations, in: Visual Languages, Proc., IEEE Symp. on, 1996, pp. 336–343. doi:10.1109/VL.1996.545307.

[13] D. Keim, G. Andrienko, J.-D. Fekete, C. Görg, J. Kohlhammer, G. Melançon, Visual analytics: Definition, process, and challenges, in: A. Kerren, J. Stasko, J.-D. Fekete, C. North (Eds.), Information Visualization, Vol. 4950 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2008, pp. 154–175. doi:10.1007/978-3-540-70956-5_7.

[14] T. Munzner, A nested model for visualization design and validation, IEEE Trans. on Visualization and Computer Graphics 15 (6) (2009) 921–928. doi:10.1109/TVCG.2009.111.

[15] T. Munzner, Visualization Analysis & Design, A K Peters Visualization, CRC Press, 2014.

[16] H. Purchase, Experimental Human-Computer Interaction: A Practical Guide With Visual Examples, Cambridge University Press, 2012.

[17] D. Archambault, T. Munzner, D. Auber, Grouseflocks: Steerable exploration of graph hierarchy space, Visualization and Computer Graphics, IEEE Trans. on 14 (4) (2008) 900–913. doi:10.1109/TVCG.2008.34.

[18] D. Auber, Y. Chiricota, F. Jourdan, G. Melancon, Multiscale visualization of small world networks, in: IEEE Symp. on Information Visualization, 2003, pp. 75–81. doi:10.1109/INFVIS.2003.1249011.

[19] L. Shi, N. Cao, S. Liu, W. Qian, L. Tan, G. Wang, J. Sun, C.-Y. Lin, Himap: Adaptive visualization of large-scale online social networks, in: IEEE Pacific Visualization Symp., 2009, pp. 41–48. doi:10.1109/PACIFICVIS.2009.4906836.

[20] M. L. Huang, Q. V. Nguyen, A space efficient clustered visualization of large graphs, in: $4^{th}$ Int. Conf. on Image and Graphics (ICIG), 2007, pp. 920–927. doi:10.1109/ICIG.2007.10.

[21] W. Didimo, F. Montecchiani, Fast layout computation of clustered networks: Algorithmic advances and experimental analysis, Information Sciences 260 (2014) 185–199. doi:10.1016/j.ins.2013.09.048.

[22] C. Muelder, K.-L. Ma, Rapid graph layout using space filling curves, Visualization and Computer Graphics, IEEE Trans. on 14 (6) (2008) 1301–1308. doi:10.1109/TVCG.2008.158.

[23] G. M. Morton, A computer oriented geodetic data base and a new technique in file sequencing, Tech. rep., IBM Ltd. (1966).

[24] M. Ghoniem, J.-D. Fekete, P. Castagliola, On the readability of graphs using node-link and matrix-based representations: A controlled experiment and statistical analysis, Information Visualization 4 (2) (2005) 114–135. doi:10.1057/palgrave.ivs.9500092.

[25] D. Auber, C. Huet, A. Lambert, B. Renoust, A. Sallaberry, A. Saulnier, Gospermap: Using a gosper curve for laying out hierarchical data, IEEE Trans. on Visualization and Computer Graphics 19 (11) (2013) 1820–1832. doi:10.1109/TVCG.2013.91.

[26] D. A. Keim, Pixel-oriented visualization techniques for exploring very large data bases, Journal of Computational and Graphical Statistics 5 (1) (1996) 58–77. doi:10.1080/10618600.1996.10474695.

[27] F. Duarte, F. Sikansi, F. Fatore, S. Fadel, F. Paulovich, Nmap: A novel neighborhood preservation space-filling algorithm, IEEE Trans. on Visualization and Computer Graphics 20 (12) (2014) 2063–2071. doi:10.1109/TVCG.2014.2346276.

[28] J. J. Van Wijk, H. Van de Wetering, Cushion treemaps: visualization of hierarchical information, in: Proc. 1999 IEEE Symp. on Information Visualization (InfoVis'99), 1999, pp. 73–78. doi:10.1109/INFVIS.1999.801860.

[29] T. Schreck, D. Keim, F. Mansmann, Regular treemap layouts for visual analysis of hierarchical data, in: Spring Conf. on Computer Graphics (SCCG'2006), 2006, pp. 184–191. doi:10.1145/2602161.2602183.

[30] J.-D. Fekete, C. Plaisant, Interactive information visualization of a million items, in: Information Visualization, 2002. INFOVIS 2002. IEEE Symp. on, 2002, pp. 117–124. doi:10.1109/INFVIS.2002.1173156.

[31] K. Stein, R. Wegener, C. Schlieder, Pixel-oriented visualization of change in social networks, in: Int. Conf. on Advances in Social Networks Analysis and Mining, 2010, pp. 233–240. doi:10.1109/ASONAM.2010.18.

[32] R. Gove, N. Gramsky, R. Kirby, E. Sefer, A. Sopan, C. Dunne, B. Shneiderman, M. Taieb-Maimon, Netvisia: Heat map and matrix visualization of dynamic social network statistics and content, in: IEEE $3^{rd}$ Int. Conf. on Social Computing (SocialCom), 2011, pp. 19–26. doi:10.1109/PASSAT/SocialCom.2011.216.

[33] M. Zinsmaier, U. Brandes, O. Deussen, H. Strobelt, Interactive level-of-detail rendering of large graphs, IEEE Trans. on Visualization and Computer Graphics 18 (12) (2012) 2486–2495. doi:10.1109/TVCG.2012.238.

[34] A. Perrot, D. Auber, Cornac: Tackling huge graph visualization with big data infrastructure, IEEE Trans. on Big Data 6 (1) (2020) 80–92. doi:10.1109/TBDATA.2018.2869165.

[35] H. Meyerhenke, M. Nöllenburg, C. Schulz, Drawing large graphs by multilevel maxent-stress optimization, in: E. Di Giacomo, A. Lubiw (Eds.), Graph Drawing and Network Visualization, Springer International Publishing, Cham, 2015, pp. 30–43. doi:10.1007/978-3-319-27261-0_3.

[36] M. Ortmann, M. Klimenta, U. Brandes, A sparse stress model, J. of Graph Algorithms and Applications 21 (5) (2017) 791–821. doi:10.7155/jgaa.00440.

[37] A. Arleo, W. Didimo, G. Liotta, F. Montecchiani, Large graph visualizations using a distributed computing platform, Information Sciences 381 (2017) 124–141. doi:10.1016/j.ins.2016.11.012.

[38] A. Arleo, W. Didimo, G. Liotta, F. Montecchiani, A distributed multilevel force-directed algorithm, IEEE Trans. on Parallel & Distributed Systems 30 (04) (2019) 754–765. doi:10.1109/TPDS.2018.2869805.

[39] G. Di-Battista, P. Eades, R. Tamassia, I.-G. Tollis, Graph drawing – Algorithms for the visualization of graphs, Prentice-Hall, 1999.

[40] Y. Frishman, A. Tal, Online dynamic graph drawing, in: Proc. of the $9^{th}$ Joint Eurographics / IEEE VGTC Conf. on Visualization (Eurovis), Eurographics Association, 2007, pp. 75–82. doi:10.2312/VisSym/EuroVis07/075-082.

[41] T. Itoh, C. Muelder, K.-L. Ma, J. Sese, A hybrid space-filling and force-directed layout method for visualizing multiple-category graphs, in: Visualization Symp., 2009. PacificVis '09. IEEE Pacific, 2009, pp. 121–128. doi:10.1109/PACIFICVIS.2009.4906846.

[42] B. Nick, C. Lee, P. Cunningham, U. Brandes, Simmelian backbones: Amplifying hidden homophily in facebook networks, in: Advances in Social Networks Analysis and Mining (ASONAM), 2013 IEEE/ACM International Conf. on, 2013, pp. 525–532. doi:10.1145/2492517.2492569.

[43] A. Nocaj, M. Ortmann, U. Brandes, Untangling the hairballs of multi-centered, small-world online social media networks, Journal of Graph Algorithms and Applications 19 (2) (2015) 595–618. doi:10.7155/jgaa.00370.

[44] S. Fortunato, Community detection in graphs, Physics Reports 486 (3–5) (2010) 75 – 174. doi:http://dx.doi.org/10.1016/j.physrep.2009.11.002.

[45] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, Journal of Statistical Mechanics: Theory and Experiment 2008 (10) (2008) P10008. doi:10.1088/1742-5468/2008/10/P10008.

[46] S. Hachul, M. Jünger, Drawing large graphs with a potential-field-based multilevel algorithm, in: J. Pach (Ed.), Graph Drawing, Vol. 3383 of LNCS, Springer, 2005, pp. 285–295. doi:10.1007/978-3-540-31843-9_29.

[47] S. G. Kobourov, Spring embedders and force directed graph drawing algorithms, CoRR abs/1201.3011 (2012).

[48] S. Hachul, M. Jünger, Large-graph layout algorithms at work: An experimental study, Journal of Graph Algorithms and Applications 11 (2) (2007) 345–369. doi:10.7155/jgaa.00150.

[49] D. Archambault, T. Munzner, D. Auber, Topolayout: Multilevel graph layout by topological features, IEEE Trans. on Visualization and Computer Graphics 13 (2) (2007) 305–317. doi:10.1109/TVCG.2007.46.

[50] J. Bentley, Multidimensional binary search trees used for associative searching, Commun. ACM 18 (9) (1975) 509–517. doi:10.1145/361002.361007.

[51] Y. Hu, L. Shi, Q. Liu, A coloring algorithm for disambiguating graph and map drawings, IEEE Trans. on Visualization and Computer Graphics 25 (2) (2019) 1321–1335. doi:10.1109/TVCG.2018.2798631.

[52] M. Harrower, C. A. Brewer, Colorbrewer.org: An online tool for selecting colour schemes for maps, The Cartographic Journal 40 (1) (2003) 27–37. doi:10.1179/000870403235002042.

[53] L. Wang, F. Du, H. P. Dai, Y. X. Sun, Random pseudofractal scale-free networks with small-world effect, The European Physical Journal B - Condensed Matter and Complex Systems 53 (3) (2006) 361–366. doi:10.1140/epjb/e2006-00389-0.

[54] J. Leskovec, A. Krevl, SNAP Datasets: Stanford large network dataset collection, `http://snap.stanford.edu/data` (Jun. 2014).

[55] B. Klimt, Y. Yang, The Enron corpus: A new dataset for email classification research, in: Proc. of the $15^{th}$ European Conf. on Machine Learning, Springer Berlin Heidelberg, 2004, pp. 217–226. doi:10.1007/978-3-540-30115-8_22.

[56] J. Sansen, R. Bourqui, B. Pinaud, H. Purchase, Edge Visual Encodings in Matrix-Based Diagrams, in: $19^{th}$ International Conf. on Information Visualisation (IV), 2015, pp. 62–67. doi:10.1109/iV.2015.22.

[57] D. Archambault, H. C. Purchase, On the effective visualisation of dynamic attribute cascades, Information Visualization 15 (1) (2016) 51–63. doi:10.1177/1473871615576758.

[58] D. Auber, D. Archambault, R. Bourqui, M. Delest, J. Dubois, A. Lambert, P. Mary, M. Mathiaut, G. Mélançon, B. Pinaud, B. Renoust, J. Vallet, TULIP 5, in: R. Alhajj, J. Rokne (Eds.), Encyclopedia of Social Network Analysis and Mining, Springer, 2017, pp. 1–28. doi:10.1007/978-1-4614-7163-9_315-1.

[59] D. Archambault, T. Munzner, D. Auber, TugGraph: Path-preserving hierarchies for browsing proximity and paths in graphs, in: IEEE Pacific Visualization Symp., 2009, pp. 113–120. doi:10.1109/PACIFICVIS.2009.4906845.

[60] T. Moscovich, F. Chevalier, N. Henry, E. Pietriga, J.-D. Fekete, Topology-aware navigation in large networks, in: Proc. of the SIGCHI Conf. on Human Factors in Computing Systems, CHI '09, ACM, New York, NY, USA, 2009, pp. 2319–2328. doi:10.1145/1518701.1519056.

[61] S. Rufiange, M. J. McGuffin, C. P. Fuhrman, Treematrix: A hybrid visualization of compound graphs, Computer Graphics Forum 31 (1) (2012) 89–101. doi:10.1111/j.1467-8659.2011.02087.x.

[62] B. Shneiderman, Extreme visualization: Squeezing a billion records into a million pixels, in: Proc. ACM SIGMOD Int. Conf. on Management of Data, SIGMOD '08, ACM, 2008, pp. 3–12. doi:10.1145/1376616.1376618.