Interactive Visualization of Small World Graphs

Frank van Ham[*]   Jarke J. van Wijk[†]

Department of Mathematics and Computer Science
Technische Universiteit Eindhoven
P.O. Box 513, 5600 MB Eindhoven
The Netherlands

ABSTRACT

Many real world graphs have small world characteristics, that is, they have a small diameter compared to the number of nodes and exhibit a local cluster structure. Examples are social networks, software structures, bibliographic references and biological neural nets. Their high connectivity makes both finding a pleasing layout and a suitable clustering hard. In this paper we present a method to create scalable, interactive visualizations of small world graphs, allowing the user to inspect local clusters while maintaining a global overview of the entire structure. The visualization method uses a combination of both semantical and geometrical distortions, while the layout is generated by a spring embedder algorithm using a recently developed force model. We use a cross referenced database of 500 artists as a running example.

CR Categories: H.5.2 [Information Interfaces and Presentation]: User Interfaces—Graphical User Interfaces; H.2.8 [Database Management]: Database Applications—Data Mining I.5.5 [Pattern Recognition]: Clustering—Algorithms

Keywords: Graph Visualization, Graph Drawing, Clustering, Small World Graphs

1 INTRODUCTION

Networks, or graphs as they are known in mathematics, form a large part of modern day information structures. From gene maps and circuit design to large cross-referenced document collections, networks play an important role in modeling relations between discrete items. Apart from local properties such as explicit relationships between two items, users dealing with these networks are often also interested in the global properties of the network: Are there any distinct groups of items that are strongly interconnected (i.e. graph clusters)? How do these split into separate clusters and how do these clusters relate? One way to answer these questions is to provide a picture of the network in which these global properties emerge visually.

In this paper we will focus on *small world graphs*. First identified by [20] in social networks, small world graphs are graphs which have a *small average path length* (average shortest path between nodes) compared to their number of nodes, but have a *high degree of clustering* compared to a random graph of the same size. The degree of clustering can be expressed by the clustering index [35, 2]. The small world property has been identified in many real world graphs such as social networks, neural networks, software systems, power grids, cross referenced knowledge bases and the in-

[*]e-mail: fvham@win.tue.nl
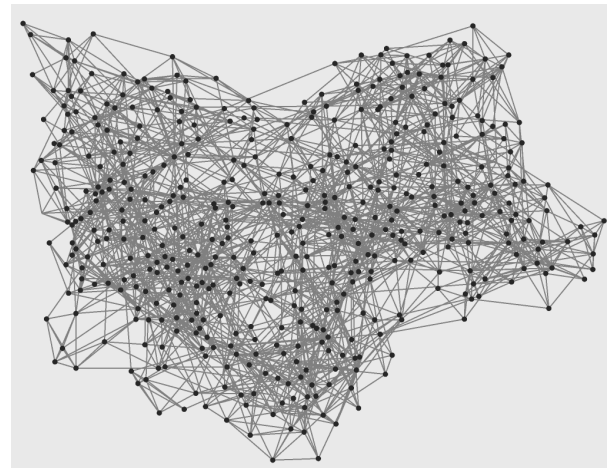
[†]e-mail:vanwijk@win.tue.nl

Figure 1: Force directed rendering of a 500 node graph, rendered with GEM

ternet. Effective visualization of these graphs can provide valuable insight in their structure.

As a running example throughout this paper we use a graph of 500 cross-referenced painters and sculptors, taken from *The Art Book* [25]. References between artists were made based on time periods and artistic movements, but also on the style and feeling of their work. The resulting connected graph consists of 500 nodes and 2486 edges. The graph has an average path length of 4 and a clustering index of 0.18. A random graph of similar size has a clustering index of 0.0093. We therefore consider it to have small world characteristics. The only class of layout algorithms that are general enough to deal with small world graphs are force directed algorithms, which model a graph as a system of repelling particles and springs. The situation in which the energy in such a system is minimal constitutes an optimal drawing. Force directed algorithms can provide satisfactory results for many types of graphs.

However, they often produce disappointing results when it comes to dense, highly connected graphs in general and small world graphs in particular. Figure 1 shows a layout for out artists graph, generated with GEM [10], a typical example of a force directed graph drawing package. Based on the semantics of the input data and the relatively high clustering index, we would like the visualization to clearly display any strongly connected subgraphs that might correspond to artistic movements. Yet we observe that the nodes are distributed rather uniformly over the image, and that the large number of overlapping edges makes it hard to draw conclusions on the global structure.

In this paper we present a new method to visualize small world graphs, such that insight in the structure can be obtained more eas-
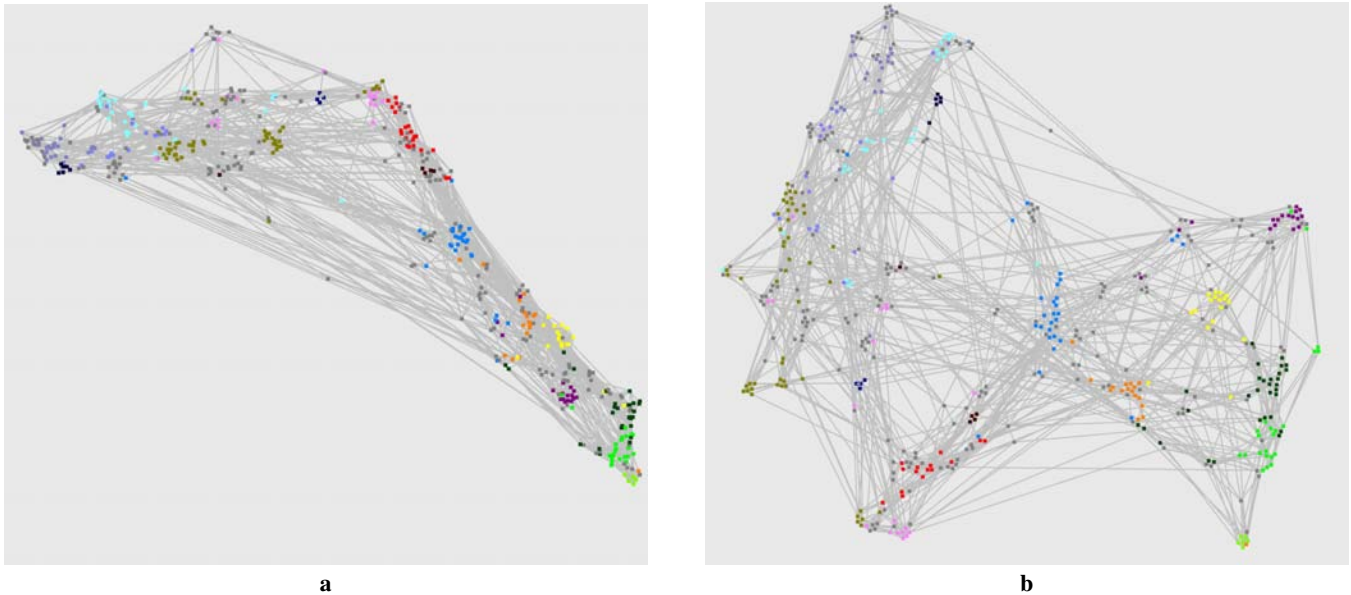
Figure 2: Layouts with random initialization: **(a)** LinLog force model ($P = -381705, M = 1500$) and **(b)** Our force model ($P = -544602, M = 1500, t_0 = 0.5, t_1 = 0.6, r_{start} = 2$). Nodes are colored based on artistic movement.

ily. Section 2 discusses related work. In section 3 we describe how the layout can be improved by employing a force model based on recent work by Noack [24]. In section 4 we show how we can reduce the number of screen items and still provide a detailed impression of the area we are interested in. Finally, in section 5 we draw conclusions and discuss future work.

2 RELATED WORK

One way to show the structure in graphs is to determine graph clusters (loosely defined as highly connected subgraphs) a priori and present these to the user. Graph partitioning has attracted much interest, from diverse areas such as data mining, VLSI design and finite element simulations. Prior literature is substantial and is spread out over many different application areas [1, 27, 7, 17, 32]. However, a hard problem with the use of any automatic clustering method is the fuzzyness of the problem itself. Although the concept of a graph cluster is easy to understand intuitively, there is no full-proof formal definition of the concept of a natural graph cluster for every task the user wants to perform. Also, graph partitioning methods generally try to optimally divide the graph into a predefined number of partitions, but in the context of information visualization it is generally not desirable (or even possible) to specify this number in advance. Other methods, most notably those based on [22] avoid this pitfall but suffer from $O(N^3)$ complexity.

A second problem is that having a decent clustering does not absolve us from constructing a consistent visualization of this clustered graph afterwards. We therefore take an alternate approach by first constructing a layout that reflects the cluster structure and basing the clustering on this layout.

The visual presentation of graphs has been studied intensively in the Graph Drawing [4] and Information Visualization [16] communities. In both fields, a popular method to lay out graphs is the force directed algorithm (FDA), introduced by Eades [8]. FDAs are widely used because they are simple to understand and implement, yet they can provide a pleasing layout for sparse graphs, showing global graph structure and graph automorphisms. A disadvantage of the basic algorithm is its computational complexity. Because in

each iteration repulsive forces have to be calculated between every node pair and the total number of iterations is typically in the order of the number of nodes N, the total complexity is $O(N^3)$. Most of the research in this area focuses on improving this [15, 26, 34, 13]. Other research on FDA's tries to create layouts that are more aesthetically pleasing by introducing different force models. A recent promising direction is the introduction of a force model that maps the connectivity between two sets of nodes to geometric distance between those sets [24]. This results in highly connected groups of nodes being displayed as areas with a higher node density.

Recently, [2] tried to visualize small world graphs by computing an edge strength metric for each edge in the graph. The graph is then partitioned into subgraphs by removing all edges that have an edge strength below a given threshold. Although initial results are promising, this method was not very effective on our sample graph. We believe this is due to the rather small clustering index of our graph (0.18), whereas [2] used graphs with very high clustering indices (mostly over 0.90).

Other methods to obtain insight into large cluttered structures are Information Galaxies, Information Landscapes [36] and the related Graph Splatting method [33]. Galaxies and Information Landscapes are mostly used to visualize large document collections. Documents are represented as multidimensional points, clustered and subsequently projected onto a two dimensional display by multidimensional scaling. Galaxies plot this two dimensional space as a series of points, but tend to suffer from information overload. Information Landscapes improve on this by constructing a smooth three dimensional height surface, where the height indicates the relative concentration of themes in an area. Graph Splatting is a related technique that creates a height field by convolving node positions obtained from a force directed layout with Gaussian splats. These methods however, offer a continuous representation of something that is inherently discrete, and although nodes can be plotted on the resulting height field, the relationships between nodes are often lost.
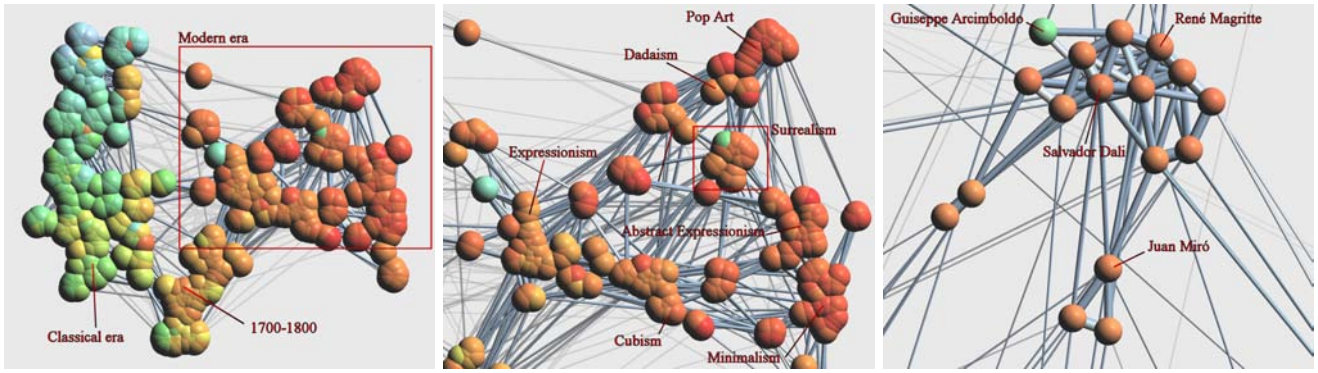
Figure 3: Using spheres to emphasize clusters. The node screen size is kept constant when zooming in such that clusters fall apart in closeups.

3 FORCE MODEL

Assume an undirected graph $G = (V, E)$ where V is the collection of nodes and E a collection of edges. We indicate an edge connecting node i and node j by e_{ij}, having weight $w(e_{ij}) = 1$. Force directed algorithms model this graph as a physical system and then try to find positions $\mathbf{p}_i$ for all nodes i such that the total energy in the system is minimal. The force on a vertex depends on the attractive force exerted by its incident edges and the repulsive force exerted by other nodes in the graph. Both forces are functions of the distance between nodes. We can write the total force exerted on a vertex i as

$$F_i = \sum_{e_{ij} \in E} f(p_{ij}) \bar{\mathbf{p}}_{ij} - \sum_{i \neq j} g(p_{ij}) \bar{\mathbf{p}}_{ij}$$

with $p_{ij} = |\mathbf{p}_i - \mathbf{p}_j|$ and $\bar{\mathbf{p}}_{ij} = (\mathbf{p}_i - \mathbf{p}_j)/p_{ij}$. The functions $f(x)$ and $g(x)$, which determine the attractive and repulsive forces respectively, typically follow physical analogies. For $f(x)$ this is usually Hooke's spring law:

$$f(x) = A \cdot (x - x_0)$$

where x_0 signifies the zero energy length of the spring. The repulsive forces generally follow an inverse square law inspired by electrostatic fields:

$$g(x) = \frac{B}{x^2}$$

Here, A and B are constants representing the strength of the attraction and repulsion respectively. The total potential energy of the system can then be expressed as

$$P = \frac{A}{2} \sum_{e_{ij} \in E} (p_{ij} - x_0)^2 - B \cdot \sum_{i \neq j} \frac{1}{p_{ij}}$$

In the optimal node configuration the potential energy of the system is minimal. Finding this global minimum directly is often not feasible, due to the high dimensionality of the problem, hence an iterative approach has to be used. A commonly used method is optimizing by steepest descent. Starting from a random configuration, the nodes are moved in the direction of the forces exerted on them, such that in the end the total force on each node is zero, or, in other words, a minimum energy state is reached. Often this will be only a local minimum, but usually the result is visually pleasant enough.

Conventional force models minimize the total variance in edge lengths, attempting to keep *all* edge lengths close to p_0. For dense graphs with a small diameter (such as small world graphs) this results in a uniform node distribution, because all nodes are kept close to their neighbors. Instead, force models that position tightly coupled groups of nodes closely together and loosely coupled groups

of nodes far apart may provide much better results. Noack derives a class of force models for which this property holds, the so-called r-PolyLog energy models. In these models the total potential energy is defined as

$$P = \sum_{e_{ij} \in E} (p_{ij} - x_0)^r - \sum_{i \neq j} ln(p_{ij})$$

Note that 3-PolyLog has an energy function that is equal to the one used by [11]. The most interesting in this respect however is the 1-PolyLog model, that has associated force functions

$$f(x) = 1$$

and

$$g(x) = \frac{1}{x}$$

Noack proves that by using the 1-PolyLog (which he dubs LinLog) model we obtain an embedding in which the distance between two clusters of nodes is inversely proportional to their coupling. The coupling between two clusters C_1 and C_2 is a measure of their connectivity and is defined as $E(C_1, C_2)/|C_1||C_2|$, where $E(C_1, C_2)$ indicates the total number of edges connecting nodes in C_1 and C_2. Using a constant force attraction might seem strange at first, but it allows intra-cluster edges to grow while inter-cluster edges shrink. In [24] some results on artificially generated graphs with high clustering indices are provided.

Experimental results on real world graphs using the LinLog energy model show that it is more susceptible to getting stuck in a local minimum than conventional methods when started from a random initial node configuration. Although this is hard to prove in general, we can reason that the force exerted on a node connected by a single long edge to the rest of the graph is much less in the LinLog model than in the 2-PolyLog model. This makes it easier for the combined repulsive forces from other nodes to prevent this node from reaching its optimal location. A better solution is not to use a random layout as a starting point but first create an acceptable layout with another force model. Generalizing this we used a r-PolyLog force model at step m out of a total of M steps ($0 \leq m < M - 1$) of the optimization process, where

$$
\begin{array}{lll}
r = r_{start} & \textbf{if} & 0 \leq m < t_1 M \\
r = \alpha_m r_{start} + (1 - \alpha_m) \cdot 1 & \textbf{if} & t_1 M \leq m < t_2 M \\
r = 1 & \textbf{if} & t_2 \leq m < M
\end{array}
$$

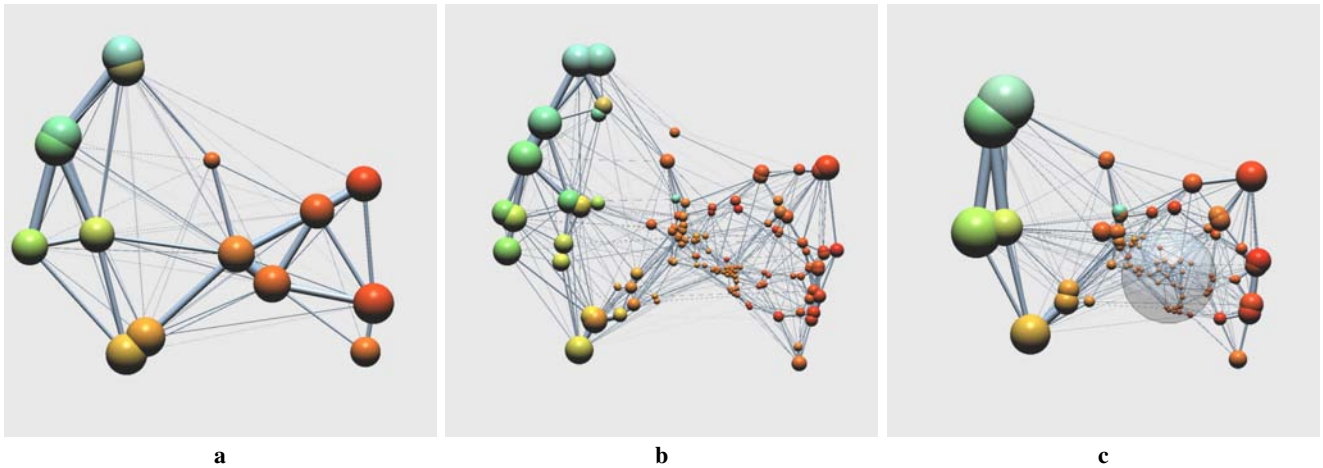with $0 \leq t_1 < t_2 < 1$, $r_{start} \geq 2$ and $\alpha_m = \frac{t_2 M - m}{t_2 M - t_1 M}$.

Figure 4: Using explicit clustering to abstract the graph: (**a**) Straight cut, (**b**) Semantic Fisheye using a cubic DOA function and (**c**) Semantic Fisheye with a linear DOA function integrated with a geometrical fisheye

This resulted in layouts with a substantial smaller energy than the ones we obtained starting from a random initial positioning in the same number of iterations, as is shown in figure 2. To test whether the visual clusters generated by the layout algorithm, actually correspond to semantical clusters in the dataset, we added expert data on a painters artistic movement (unfortunately we were unable to retrieve this information for all painters) and used it to color the nodes (Figure 2b). For all modern movements (right half) the image shows a clear coherence between the movement and the clustering in the layout. For most classical painters (left half) this coherence is much less clear however. Although we are able to find some small structural clusters based on movement, we suspect that most pre-17th century artists are likely clustered based on another (unknown) property, such as geographical location.

4 VISUALIZATION

Given a layout L_G of a graph G, we can state that highly connected subgraphs are now visible as areas with a higher node density. However, the image shown in figure 2(b) is still noisy and hard to read. Apart from that, displaying all nodes and edges at one single time is simply not an option if when dealing with larger graphs. We have to provide a way to reduce the number of visible elements, while maintaining the global structure of the graph. In other words, we have to provide a visual abstraction of L_G. At the same time the user will also be interested in details, so we have to find an acceptable way to integrate detail with context. The visualization of edges also remains a problem, since the large number of long edges in the picture make it hard to trace individual connections. We discuss each of the problems mentioned above in the next sections

4.1 Visual Abstraction

A simple but effective way to provide visual abstraction without losing a sense of the global structure is to render the nodes as overlapping spheres with constant size in screen space. The resulting configuration of partly overlapping spheres visually abstracts a cluster of nodes, showing it as a blob like structure. At the same time, the internal structure of a cluster is still visible as a shaded Voronoi diagram, giving an impression on how the cluster falls apart when we zoom in. By keeping the screen size of nodes constant while zooming in on the structure, we obtain progressively more details. In fact, this method can be seen as a fast alternative to

[30], with the added ability to continuously display the graph at any level of abstraction by zooming in or out. Figure 3 shows our sample graph at three zoom levels, nodes are colored based on birthdate. For example, 16th century painter Guiseppe Arcimboldo is classified as (a very early) surrealist, which is not so strange considering his paintings.

Another often used way to provide abstraction is to compute a clustering on L_G. In the layout the distance between clusters is inversely proportional to their coupling. This means that we can use the geometric distances between nodes as a useful clustering metric, since the most tightly coupled clusters are geometrically closest. We perform a bottom up agglomerative clustering on G using any distance measure $d(i, j)$ between two clusters i and j. Define a cluster k as a tuple $(\mathbf{c}_k, r_k, d_k, f_k)$, where $\mathbf{c}_k$ represents the cluster's center, r_k its radius, d_k the distance between its two subclusters and f_k its parent cluster. Initially we assign all nodes i to separate clusters $(\mathbf{p}_i, r_{base}, 0, nil)$. Next, we iteratively merge clusters by selecting the two clusters i and j with minimal $d(i, j)$ and merging them in a new cluster k. For all edges $e_{(ij)x}$ incident to i or j we create a new edge e_{kx}. If an edge e_{kx} already exists we increase its weight by $w(e_{(ij)x})$.

The new cluster has to provide some visual feedback on the amount of leaf nodes it contains and their approximate positions. Let $LN(k)$ be the set of leaf nodes in cluster k. A straightforward choice for $\mathbf{c}_k$ is the average position of all leaf nodes $l \in LN(k)$. We let the radius r_k depend on $|LN(k)|$. Linear (radius is proportional to size), a square root (area is proportional to size) or a cube root (area is proportional to volume) function are all candidates. We found that the square root function provides better results in general, since a linear function sometimes leads to excessively large clusters, while a cube root function makes it too hard to discern size differences.

This leaves us with choosing a suitable distance function $d(i, j)$ to determine the distance between two clusters. Widely used metrics in clustering literature define the inter cluster distance as a function of the distances between their (leaf)members [14].

Well known variants are minimum link, maximum link and average link distance. Minimum link defines the distance $d(i, j)$ between clusters i and j as the distance between the two closest members of both clusters, maximum link uses the distance between the

two furthest members, while average link uses the average distance between all members.

Clustering using the minimum link measure has the tendency to form long chains, while maximum link clustering is very sensitive to outliers. Average link clustering generally provides a good optimum between both and is our metric of choice here. For the distances between cluster members (i.e. the leaf nodes) we can choose either the geometric distance or the edge length, with the edge length between unconnected nodes set to infinity. The latter method does not allow clusters that are not interconnected but positioned geometrically close by a suboptimal layout to be merged. Although aesthetically less pleasing, we use the edge lengths for clustering because it creates better clusters.

Applying the cluster algorithm outlined above results in a **binary hierarchy** or dendrogram, where the level in the hierarchy for a cluster k is equal to the distance d_k between its child clusters. A simple way to create abstractions for a clustered graph is to define a global *degree of abstraction DOA* $\in [0,1]$ and display only those clusters k for which $d_k \leq d_{root} \cdot DOA < d_{f_k}$ (Figure 4.1). To create a continuous transition when *DOA* is changed we can interpolate between the size and position of cluster k and the size and position of its parent cluster using a local parameter λ_k where

$$\lambda_k = ((DOA \cdot d_{root}) - d_k)/(d_{f_k} - d_k)$$

λ_k is a parameter that indicates the amount of interpolation between the position and size of a cluster k and the position and size of its parent f_k based on the degree of abstraction at that point. We can state that $0 \leq \lambda_k \leq 1$ for all visible clusters, because $d_k \leq d_{root} \cdot DOA < d_{f_k}$.

Based on user preference we can either use linear, exponential or slow-in slow-out interpolation (Fig 5). We found a combination of a linear interpolation for the positions and an exponential interpolation for the cluster size to give the best results. Interpolating cluster sizes exponentially prevents small isolated clusters from growing to their parents size too quickly. Using λ we can compute sizes and positions for all clusters visible in the current 'slice' of the hierarchy and we can show the graph at an arbitrary level of detail by varying the DOA. Note that a similar idea was coined by [26], but their 'horizons' are defined on a hierarchy with discrete levels and the hierarchy is generated based on a simple recursive space subdivision technique that does not always generate very accurate clusters. The procedure outlined above is in effect a continuous version of this idea. Figure 4a shows our artists graph at a higher level of abstraction, representing a reduction of 97% in the number of visible nodes.
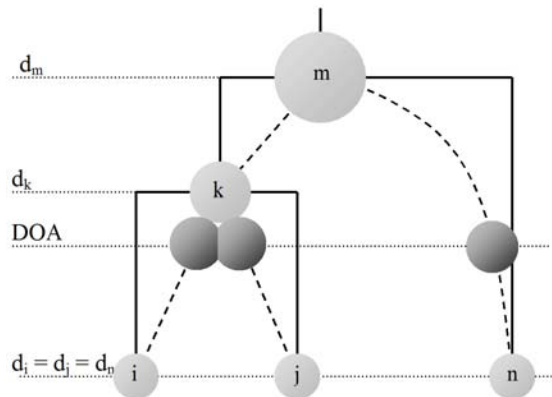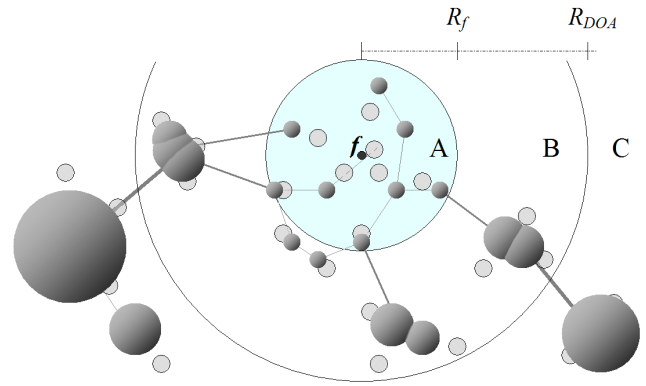

Figure 6: Top down view of visualization area indicating the three different methods of visual abstraction: area A uses a fisheye lens to distort node positions, area B incrementally abstracts nodes and area C displays nodes with a constant DOA to avoid unnecessary motion in the periphery
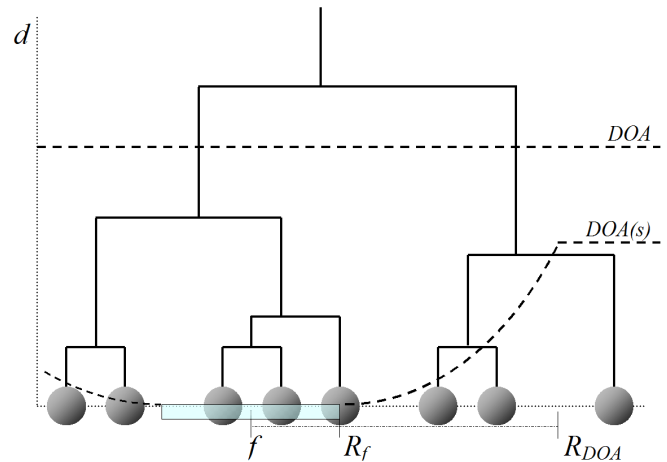

Figure 7: Schematic cross section, showing the cluster dendrogram with constant and variable DOA functions

Figure 5: Interpolation of sizes and positions based on current DOA. Linear interpolation on the left, exponential interpolation on the right

4.2 Detail and Context

Providing both detailed information as well as a global context in one image is one of the fundamental problems in Information Visualization. Although we are able to provide views of a graph at different levels of abstraction, the user will probably be more interested in obtaining detail information for some specific section of the graph. One solution is to embed this detail in the global graph structure in a useful way, allowing the user to maintain both a context for the local area he is interested in, as well as a consistent global mental map of the entire structure.

One often used way of embedding local detail in global context is the Fisheye View. Generalized Fisheye views [12] define a Degree of Interest for an information element based on its a priori importance and its distance to a focal point. Most applications of the fisheye technique fall into one of two categories. One are the *distortion oriented* techniques, which distort the sizes and positions
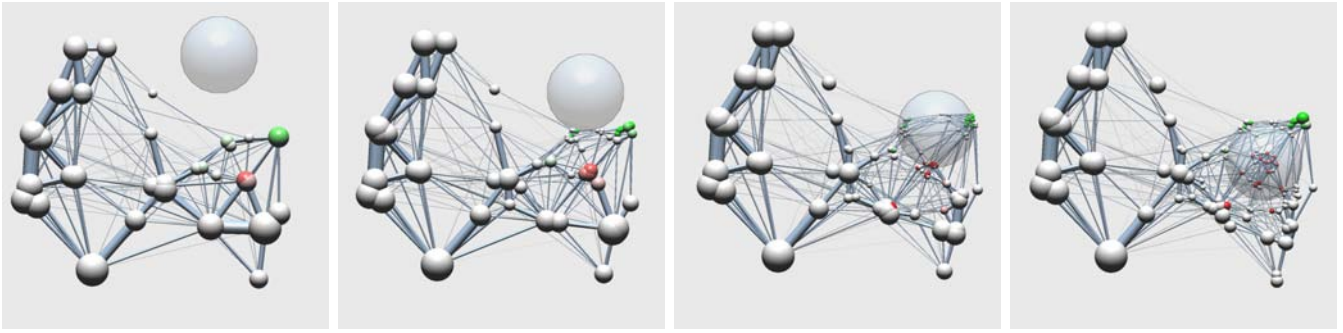
Figure 8: Series of frames showing the effect of a changing focus on the layout. Surrealist artists are indicated in red, artists belonging to the Pop-Art movement in green.

of nodes based on their geometric distance to the focus. This can be either done in the layout stage by employing a hyperbolic distance measure [18, 21] or as a successive projection step on an existing layout [28, 19, 6, 31]. The other are *data suppression* techniques that hide or abstract nodes based on the distance to a focus, creating an abstracted layout. Although conceptually closer to Furnas' original idea, relatively few researchers have pursued this path. Most notable are [9, 26], while others, such as [29] also use abstraction coupled with a distortion function but require the user to manually expand and collapse clusters of interest.

Distortion oriented techniques provide a local geometric zoom that is easily understandable, but still require all nodes to be drawn. Data suppression techniques avoid this and only display a fraction of the nodes, but they require extra attention to keep the relationship between nodes and their abstractions understandable. Often used ways to do this are by visual containment (i.e. always displaying a node visually contained in its abstraction) or animation (by animating from an abstracted layout to a detailed layout). Since both techniques have their individual merits, we propose to integrate them in a single scheme (fig 4c).

That is, we use a geometric fisheye by varying the local zoom factor with the distance from a focus and a semantic fisheye by varying the DOA with the distance from the focus. We therefore divide the visualization area into three concentric sections, based on the distance s of a point $\mathbf{p}$ to a focal point $\mathbf{f}$ (Figure 6) : A focal area A where $s \le R_f$, an area B with $R_f < s \le R_{DOA}$ in which we increasingly abstract nodes and an area C in which we keep the degree of abstraction constant.

For points within the focal area A we use a variable geometric zoom. We prefer a function Z that magnifies sections close to $\mathbf{f}$ but does not distort when $s \ge R_f$. A suitable function is the Fisheye Distortion function from [28], i.e.,

$$
\begin{aligned}
Z(s) &= \frac{(z+1)s}{zs/R_f+1} &\quad \textbf{if} \quad s < R_f \\
Z(s) &= s &\quad \textbf{if} \quad s \ge R_f
\end{aligned}
$$

In this case z is a factor that determines the amount of zoom near the focus. Instead of also scaling the node sizes, like a common fisheye lens does, we keep the size of nodes in screen space constant, analogous to the abstraction method mentioned at the beginning of the previous paragraph. This gives the visual effect that a tightly packed cluster of nodes gets 'pulled apart' near the center of the fisheye lens.

For points outside the focal area we use a semantic fisheye, that is, points $\mathbf{p}$ for which $R_f < s \le R_{DOA}$ are increasingly abstracted with increasing distance from the focus (area B). In other words, instead of a single global abstraction level *DOA*, we define a function

$DOA(s) \in [0,1]$ that specifies a degree of abstraction as a function of the distance s of a point $\mathbf{p}$ to $\mathbf{f}$.

We then render the graph by only rendering the highest clusters in the hierarchy k for which

$$
d_k \le d_{root} \cdot DOA(|\mathbf{c}_k - \mathbf{f}|) \wedge d_{root} \cdot DOA(|\mathbf{c}_{f_k} - \mathbf{f}|) < d_{f_k}
$$

We can find these clusters quickly by inspecting the hierarchy in a top-down fashion. We use a DOA function that does not abstract nodes in area A, is increasing in area B and has a constant value in area C:

$$
\begin{aligned}
DOA(s) &= 0 &\quad \textbf{if} \quad s < R_f \\
DOA(s) &= a(s) &\quad \textbf{if} \quad R_f < s \le R_{DOA} \\
DOA(s) &= a(R_{DOA}) &\quad \textbf{if} \quad R_{DOA} < s
\end{aligned}
$$

The choice of an increasing function a is arbitrary, but linear and cubic functions are straightforward choices here. Figure 4 shows samples of both. By using a constant DOA for nodes in area C, we avoid changing the interpolated sizes and positions for these nodes when the focus changes. Since movement is such a strong visual cue, viewer attention might get distracted from the focal area to the context, which is undesirable. Figure 7 shows these concepts schematically.

Summarizing the above, a node i at distance $s = |\mathbf{p}_i - \mathbf{f}|$ from the focus point is displayed by:

1. **abstracting** the node based on a function $DOA(s)$.

2. **projecting** the abstracted node at position $\mathbf{f} + Z(s) \cdot \frac{\mathbf{p}_i - \mathbf{f}}{s}$.

By visually indicating the edge of the fisheye distortion, we make these two different zooming actions (one geometric and the other semantic) more understandable for the user (Fig 4c). The overall effect obtained when moving the focus bears a lot of resemblance to the Magic Lens [5]. Fig 8 shows a number of intermediate frames when moving the focus. The fact that only O(Log(N)) nodes are visible on average (depending on the choice of the *DOA* function) allows us to maintain interactive performance. We also provide a consistent, albeit abstracted, picture of the context. In full geometric distortion techniques the positions of items in the context depends on the current position of the focus. This prevents the creation of a consistent mental map.

We have also applied our method to various other graphs. As an example, figure 9 shows the largest connected part of the data set used in [33]. The nodes represent 824 papers in the proceedings of the IEEE Visualization conferences from 1990 to 2002, with edges denoting citations. We could identify several main research areas by looking at the conference session the papers appeared in.

204

4.3 Edges

So far we have only considered the visualization of nodes, this section discusses the visualization of edges. Traditionally, edges are visualized in node link diagrams as straight lines with a fixed width. As figure 2b shows, many long intersecting edges lead to a noisy image. We make the following observations:

- Line crossings can only be resolved by using the Gestalt principle of visual continuity, i.e., we assume that the direction of the lines after a crossing is the same;

- The use of screen space of edges is proportional to their length;

- The longer edges account for only a small part of the total number of edges.

Based on the first observation, we can improve visual continuity by displaying edges as shaded tubes. The shading augments the visual continuity, and also depicts edges and nodes similarly. We plot short edges closer to the viewer than long edges, resolving most intersections and de-emphasizing longer edges. As a consequence of the second observation, long edges use up a large part of the available pixels, although they are no more important than short edges. To resolve this imbalance we aim to keep the amount of space for each edge constant. However, simply keeping the product of width and length of edges constant leads to edges that are too thin, and also, we are using 3D tubes here. Therefore, we keep the volume of an edge constant.

Based on the third observation, we can state that a small portion of the edges takes up a large portion of the display space. In our case, we found that the 5% longest edges account for 29% of the total edge-length and hence use up almost one third of all pixels allocated to edges. We can decrease the total number of pixels used for the display of edges in the graph substantially by leaving out only a few percent of the longest edges. Since simply eliding these edges would cause a loss of information, we instead choose to draw a small fraction (say, 5%) of the longest edges transparently.

5 CONCLUSIONS AND FURTHER WORK

We have presented a new method that enables us to get insight in the structure of small world graphs. Small world graphs are notoriously hard to visualize because of their highly connected nature. Our method contributes:

- A modifications to a force model from [24] in order to obtain layouts that better reflect the natural cluster structure of the graph.

- A method for continuous visual abstraction that combines both explicit clustering and visual clustering. Both clustering methods provide smooth interpolation between various levels of detail.

- The combined use of the previous techniques in an interactive generalized fisheye scheme, using both semantic and geometric distortions.

The resulting visualization shows an abstracted view of the entire graph, in which a close up view is integrated. Because on average only O(Log(N)) clusters are visible, the system is fast enough to perform at interactive speeds.
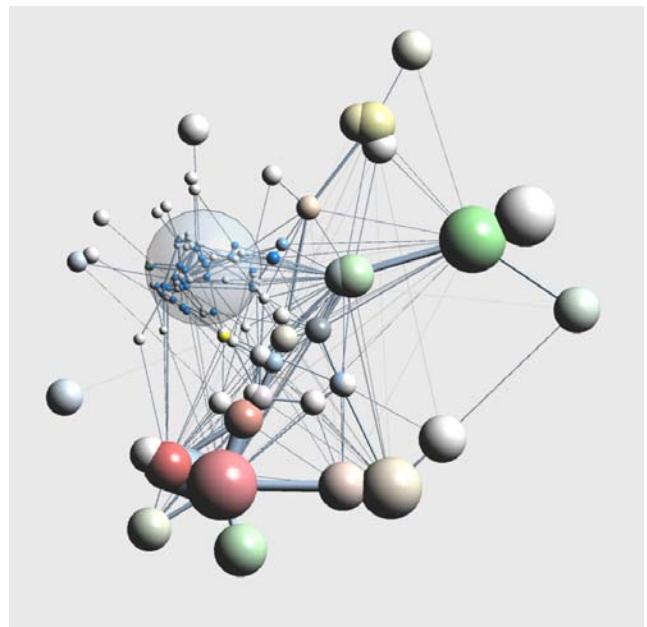

Figure 9: Citations between papers published in 12 years of IEEE Visualization conference proceedings, focus is centered on volume visualization (blue). Other significant clusters are flow visualization (red), terrain visualization/surface simplification (green) and information visualization (yellow)

An obvious addition to the visualization is the use of multiple foci. Although not implemented yet, we can simply change the current function $d(\mathbf{p}, \mathbf{f})$ to a function $d_M(\mathbf{p}, F)$ that returns the distance of $\mathbf{p}$ to the nearest focus in the collection of foci F. Another possible improvement concerns the amount of uncertainty in the visualization. The further away from the focus a cluster is, the more inaccurate its representation. The current node representation (i.e. a sphere) gives the impression of a data item with a clearly defined position and size. An alternative might be to use splats for node representation and vary the shape of the splat with increasing distance to $\mathbf{f}$. That is, we use a sphere for nodes close to $\mathbf{f}$ and a gaussian splat for nodes in the periphery. Also, automatic labelling of clusters is an important issue. Currently semantics for clusters can only be given by inspecting leafnodes inside that cluster manually. The usefulness of these visualizations would certainly be increased if we could characterize each cluster automatically.

In the future we will focus on larger graphs, consisting of tens of thousands of nodes and more. The current layout method inherits the $O(N^3)$ complexity of the standard force directed algorithm, but recent efficiency improvements for this, such as Barnes-Hut optimization [3, 26] can be used, such that a $O(N^2 \log N)$ complexity is achieved. Note that multigrid/multiscale methods such as [15], perform poorly on dense graphs with small diameters (as their authors also note). The clustering step currently has an $O(N^2 Log(N))$ complexity and may also benefit from this space subdivision technique. Recent research by [23] suggests a promising $O(N^2)$ algorithm to explicitly compute a dendrogram of clusters based on the graph structure. We intend to use this method to evaluate the quality of our layout further. Both steps (layout and clustering) however are preprocessing steps and are not critical to the interactivity, and it will be interesting to see how our method performs on larger real-world graphs.

REFERENCES

[1] Charles J. Alpert and Andrew B. Kahng. Recent directions in netlist partitioning: A survey. Technical report, Computer Science Department, University of California at Los Angeles, 1995.

[2] D. Auber, Y. Chiricota, F. Jourdan, and G. Melançon. Multiscale visualization of small world networks. In *Proc. of IEEE Symp. on Information Visualization 2003*, pages 75–81. IEEE CS Press, 2003.

[3] Joshua E. Barnes and Pier Hut. A hierarchical o(n log n) force calculation algorithm. In *Nature*, volume 324(4), pages 446–449, 1986.

[4] Guiseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. Graph drawing : Algorithms for the visualization of graphs. Prentice Hall, 1999.

[5] Eric A. Bier, Maureen C. Stone, Ken Pier, William Buxton, and Tony D. DeRose. Toolglass and magic lenses: The see-through interface. *Computer Graphics*, 27(Annual Conference Series):73–80, 1993.

[6] M. S. T. Carpendale, D. J. Cowperthwaite, F. D. Fracchia, and T. C. Shermer. Graph folding: Extending detail and context viewing into a tool for subgraph comparisons. In Franz J. Brandenburg, editor, *Proc. 3rd Int. Symp. on Graph Drawing*, pages 127–139. Springer-Verlag, 1995.

[7] F. Chung and S. T. Yau. Eigenvalues, flows and separators of graphs. In *Proc. of the 26th Annual ACM Symposium on Theory of Computing*, pages 1–8, 1997.

[8] P. Eades. A heuristic for graph drawing. In *Congressus Numerantium*, volume 42, pages 149–160, 1984.

[9] K. M. Fairchild, S. E. Poltrock, and G. W. Furnas. Semnet: Three-dimensional graphic representations of largeknowledge bases. In R. Guindon, editor, *Cognitive Science and Its Applications for Human Computer Interaction*, pages 201–233. Lawrence Erlbaum, 1988.

[10] Arne Frick, Andreas Ludwig, and Heiko Mehldau. A fast adaptive layout algorithm for undirected graphs. In Roberto Tamassia and Ioannis G. Tollis, editors, *Proc. of the DIMACS Int. Workshop on Graph Drawing*, pages 388–403. Springer-Verlag, 1994.

[11] T. M. J. Fruchterman and E.M. Reingold. Graph drawing by force-directed placement. *Software - Practice and Experience*, 21(11):1129–1164, 1991.

[12] G. W. Furnas. Generalized fisheye views. In *Human Factors in Computing Systems CHI '86 Conf. Proc.*, pages 16–23. ACM Press, 1986.

[13] P. Gajer, M. Goodrich, and S. Kobourov. A multi-dimensional approach to force-directed layouts of large graphs. In Joe Marks, editor, *Proc. 8th Int. Symp. on Graph Drawing*, pages 211–221. Springer-Verlag, 2000.

[14] J. Han and M. Kamber. *Data Mining - Concepts and Techniques*. Morgan Kaufmann, 2001.

[15] David Harel and Yehuda Koren. A fast multi-scale method for drawing large graphs. In *Graph Drawing: 8th International Symposium (GD'00)*, pages 183–196, 2000.

[16] Ivan Herman, Guy Melançon, and M. Scott Marshall. Graph visualization and navigation in information visualization. volume 6(1), pages 24–43, 2000.

[17] M. L. Huang and P. Eades. A fully interactive system for clustering and navigating large graphs. In S. Whitesides, editor, *Proc. 6th Int. Symp. on Graph Drawing*, pages 374 – 383. Springer-Verlag, 1998.

[18] J. Lamping, R. Rao, and P. Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *Human Factors in Computing Systems CHI '95 Conf. Proc*, pages 401–408. ACM Press, 1995.

[19] Y. K. Leung and M. D. Apperley. A review and taxonomy of distortion-oriented presentation techniques. *ACM Transactions on Computer-Human Interaction*, 1(2):126–160, 1994.

[20] S. Milgram. The small world problem. *Psychology Today*, 2:60–67, 1967.

[21] T. Munzner. H3 : Laying out large directed graphs in 3d hyperbolic space. In *Proc. of IEEE Symp. on Information Visualization 1997*, pages 2–10. IEEE CS Press, 1997.

[22] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E 69, 026113*, 2004.

[23] M.E.J. Newman. Fast algorithm for detecting community structure in networks. *Phys. Rev. E (in press)*, 2004.

[24] A. Noack. An energy model for visual graph clustering. In *Proc. 11th Int. Symp. on Graph Drawing*, pages 425–436. Springer-Verlag, 2003.

[25] Phaidon. *The ART Book*. Phaidon Press Limited, London, 1994.

[26] Aaron Quigley and Peter Eades. Graph drawing, clustering, and visual abstraction. In Joe Marks, editor, *Proc. 8th Int. Symp. Graph Drawing*, pages 197–210. Springer LNCS 1984, 2000.

[27] Tom Roxborough and Arunabha Sen. Graph clustering using multi-way ratio cut. In Giuseppe Di Battista, editor, *Proc. 5th Int. Symp. on Graph Drawing*, pages 291–296. Springer-Verlag, 1997.

[28] Manojit Sarkar and Marc H. Brown. Graphical fisheye views of graphs. In *Human Factors in Computing Systems CHI '92 Conf. Proc.*, pages 83–91. ACM Press, 1992.

[29] Doug Schaffer, Zhengping Zuo, Saul Greenberg, Lyn Bartram, John Dill, Shelli Dubs, and Mark Roseman. Navigating hierarchically clustered networks through fisheye and full-zoom methods. *ACM Transactions on Computer-Human Interaction*, 3(2):162–188, 1996.

[30] T. C. Sprenger, R. Brunella, and M. H. Gross. H-blob: A hierarchical visual clustering method using implicit surfaces. In *Proc. of IEEE Visualization*, pages 61–68. IEEE CS Press, 2000.

[31] M. D. Storey, F. D. Fracchia, and H. A. Muller. Customizing a fisheye view algorithm to preserve the mental map. *Journal of Visual Languages and Computing*, 10(3):245–267, 1999.

[32] Stijn van Dongen. *Graph Clustering by Flow Simulation*. PhD thesis, University of Utrecht, 2001.

[33] Robert van Liere and Wim de Leeuw. Graphsplatting: Visualizing graphs as continuous fields. In *IEEE Transactions on Visualization and Computer Graphics*, volume 9(2), pages 206–212, 2003.

[34] C. Walshaw. A multilevel algorithm for force-directed graph drawing. In Joe Marks, editor, *Proc. 8th Int. Symp. Graph Drawing*, pages 171–182. Springer-Verlag, 2000.

[35] D. J. Watts and S.H. Strogatz. Collective dynamics of 'small-world' networks. *Nature 393*, pages 440–442, 1998.

[36] J. A. Wise, J. J. Thomas, K. Pennock, and D. L. Lantrip. Visualizing the non-visual: Spatial analysis and interaction with information from text documents. In S.K. Card, J.D. Mackinlay, and B. Shneiderman, editors, *Readings in Information Visualization*, pages 442–250. Morgan Kaufmann Publishers, 1999.