HiMap: Adaptive Visualization of Large-Scale Online Social Networks

Lei Shi Nan Cao Shixia Liu Weihong Qian Li Tan
IBM China Research Laboratory

Guodong Wang
Tsinghua University

Jimeng Sun Ching-Yung Lin*
IBM T. J. Watson Research Center

ABSTRACT

Visualizing large-scale online social network is a challenging yet essential task. This paper presents HiMap, a system that visualizes it by clustered graph via hierarchical grouping and summarization. HiMap employs a novel adaptive data loading technique to accurately control the visual density of each graph view, and along with the optimized layout algorithm and the two kinds of edge bundling methods, to effectively avoid the visual clutter commonly found in previous social network visualization tools. HiMap also provides an integrated suite of interactions to allow the users to easily navigate the social map with smooth and coherent view transitions to keep their momentum. Finally, we confirm the effectiveness of HiMap algorithms through graph-travesal based evaluations.

Index Terms: H.5 [User Interfaces] Graphical User Interfaces, I.3 [Methodology and Techniques] Interaction Techniques

Keywords: adaptive visualization, clustered graph, social network visualization

1 INTRODUCTION

In the past three years, we has witnessed the tremendous growth of the online social networks. As one of the most exciting applications of the Internet, it now connects 1.45 billion registered users, more than ten times the year of 2005, according to the significant online social network list in Wikipedia [33]. Now each Internet subscriber is hosting himself in at least one social network websites in average, and these sites account for 15 out of the Alexa top 50 websites [4] around the world. People now connect friends and publish blogs in Facebook and Myspace, share movie and picture in Youtube and Flickr, and present resume and search for job in LinkedIn, shifting nearly all traditional social activities online.

While a variety of applications have been integrated into the online social network platform, there is still big gap in revealing the social structure and hierarchies through their underling connections. Facebook has launched TouchGraph [2], LinkedIn and MySpace also obtain similar applications via their open APIs, but all these tools are limited to only reveal the networks around a given user, without an overview of the entire social networks. Moreover, they faithfully present the social connections, but avoid deep information processing, such as clustering and summarizing. To present the hierarchical view of the social network is critical, since it first allows the online user to comprehend clearly his exact role and build mental impression about the entire network, and secondly, provides artifactitious yet rich material for the social network operators and analysts to investigate the network further. In the past, quite a lot stand-alone visualization systems have been built for viewing the social networks, such as Vizster [14], MatrixExplorer [15] and NodeTrix [16]. They either adopted the flat visualization that tiles all the nodes (representing the social network users) together in one view, or they introduce customized visual metaphors, such as adjacency matrix, to compose the visualization. The former designs are quite useful in showing an overview to the network, and the later ones would be popular for the serious studies focusing more on the details of the network.

In this paper, our motivation for the social network visualization is significantly different from these previous works. Our system is built upon the assumption that the entire social network may scale to millions of nodes which could turn the overview graph into an all-black copy paper; and the target users of the system are generally ordinary people that will be frustrated and impatient with the advanced visualization metaphors. The goals for our visualization system is defined below:

(*i*) Each graph view of the network should be adaptively visualized in a readable manner that is easy to be comprehended, independent with its scale, topology and the screen size to display.

(*ii*) A suite of navigation methods should be provided so that it is capable to visualize and diagnose every detail of the network.

(*iii*) Smooth animations should be presented between any view changes, so as to keep user's momentum [25].

(*iv*) The visualization system should run fast enough and keep lightweight: it could catch up with the animation speed and load social network data incrementally and on-demand.

We have designed and implemented HiMap, the Hierarchical interactive social Maps, which could adaptively visualize large-scale online social networks while meeting our goals above. We choose to use clustered node-link graph for HiMap, as shown in Figure 1 for the visualization of a group of IBM users with their social connections extracted by smallblue [23]. It is well-known that the online social network possesses highly clustered and self-similar community structure. Herein, the clustered graph, which could reveal their built-in hierarchy information, is one of the best way to visualize it, let alone it also provides semantic abstraction that makes the readable visualization possible. We have also developed the Stable Kamada-Kawai layout algorithm for Clustered graphs (SKK-C). It works in a recursive manner, aiming to avoid cluster overlapping, stabilize layout across consecutive views, while preserving the excellent graph aesthetics inherited from the original Kamada-Kawai layout algorithm [20].

To adapt each visualization view with the changing screen size, we have introduced a novel adaptive data loading method. It includes three steps: 1) Rank the visual items in the same hierarchy by pre-defined importance metric; 2) Quantify the number of displayed visual items[1] according to the current screen size; 3) Adaptively load the visual items by their rankings and the available visual item budget. We currently provide two ranking algorithms for separate usage scenarios. Although they are of relatively high computational complexity, the ranking results, as well as part of the item quantification intermediate results, could be pre-computed offline, so that our adaptive data loading method will run fast enough for online purpose.

HiMap employs the adaptive data loading method to preserve the readability of the graph view, but it does not trade-off this advantage through the compensation of losing the reachability of network

*e-mails: {shllsh,nancao,liusx,qianwh,tltan}@cn.ibm.com, wanggd04@gmail.com, {jimeng,chingyung}@us.ibm.com. This work was performed when Guodong Wang was an intern student at IBM China Research Laboratory.

[1] Visual items in this paper include the clusters, leaf-nodes and the edges connecting them.

IEEE Pacific Visualization Symposium 2009
April 20 - 23, Beijing, China
978-1-4244-4404-5/09/$25.00 ©2009 IEEE

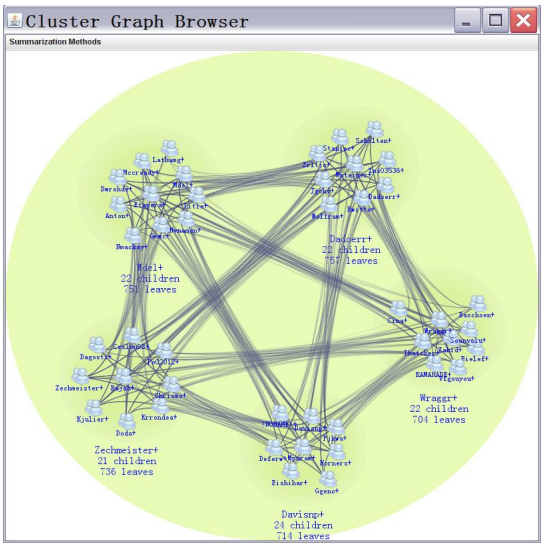


Figure 1: HiMap visualization of a group of IBM smallblue users. Edges are bundled together geometrically.

details. Instead, it is equipped with a rich set of interactions, including selecting, dragging, zooming, panning, and two classes of edge bundling methods. Within them, the zooming operation is the most unique one compared with the interaction of the other graph genres. We have implemented three kinds of zooming operations: the hierarchical zoom-in/zoom-out (namely drill-in/roll-up later in the paper) that navigate through different graph hierarchies, the semantic zoom-in/zoom-out that focus the view on a smaller/larger portion of the previous graph and adaptively reload graph data, and the traditional geometric zoom-in/zoom-out operation. To maintain the user's visual momentum, we have also designed customized animations for each HiMap interactions upon view changes.

The contribution of this paper is three-fold:

(*i*) We designed, implemented and evaluated the methods of adaptively visualizing the online social networks with clustered graph;

(*ii*) We extended the classic Kamada-Kawai layout algorithm to SKK-C algorithm that could stabilize the graphs across consecutive views and avoid cluster overlapping;

(*iii*) We customized the interactions for navigating the clustered graph visualization of online social networks and designed specific animations to keep the user's visual momentum during the navigation.

The rest of the paper is organized as follows. Section 2 summarizes related work. Section 3 overviews the HiMap visualization pipeline. Section 4 details its adaptive data loading method and introduces the ranking algorithms. Section 5 proposes the SKK-C layout algorithm, Section 6 describes the HiMap interactions and smooth animation design. Section 7 presents various evaluation results, and finally Section 8 concludes the paper.

2 Related Work

Here we summarize the related works by three categories: the social network visualization systems, including the general topic of huge graph visualization; the algorithms for visual filtering and summarization; and the works on graph drawing, especially the layout algorithms for clustered graphs.

Vizster [14] is a system for pleasant exploration of online social networks. It provides interactions for both user-oriented navigation and interactive community detection. MatrixExplorer [15] and Nodetrix [16] proposed the idea of combining the advantage of adjacency matrix and traditional node-link visualization to present the social network relationships. Van Ham and van Wijk introduced the focus+context techniques to visualize small-world graphs [31].

The clustered graph visualization is first studied in Feng's thesis [11], but her focus is on the aesthetic graph drawing methods, e.g. how to avoid edge crossings. They also considered the variant in 3D visualization form [10][18]. The approach by Auber et al. [5] is probably the most closed work to our HiMap design, which also adopts the hierarchical graph for visualization. However, it puts more emphasis on the decomposition of social network graphs, while our work discusses more on the subsequent graph visualization. For a detailed introduction on huge graph visualization works, please refer to Herman's comprehensive survey [17].

There has been considerable works on visual filtering and summarizing of huge graphs. Six and Tollis discussed several ways to visualize grouping relationships and proposed the method of interactive node abstraction [30]. The works in [22][28] studied the sampling methods to simplify the huge graph. They generally rely on randomization methods and take the user's visual focus into account. Edge summarization and bundling is another popular technique to reduce the visual clutter. Jia et al. proposed the idea of filtering edges with small betweenness centrality [19], while Van Ham and Wattenberg evaluated the approach to produce the layout of a graph with minimum edge centrality spanning tree and add other edges with arcs [32]. For edge clustering, Cui et al. have proposed the method of using some control points to bundle the edges [8]. OntoVis [29] includes a set of novel techniques for graph abstraction and filtering. It leverages the ontology graph to facilitate the analysis on social networks. Perer and Shneiderman developed an integrated system, SocialAction [27], which introduces the attribute ranking based solution to overview, filter, find outliers and code the social network visualization. Users of SocialAction is free to choose visualization metaphors to analysis and discover interesting patterns from complex network.

The layout algorithm is a well-studied problem in graph drawing community. The most popular solutions are generally energy based force-directed algorithms [20][12][9]. For clustered graph, most current approaches apply the classic force-directed solution [18][11][5], but few are based on the Kamada-Kawai layout algorithm, which is the major approach in this paper.

3 HiMap Overview

This section overviews the techniques involved in HiMap design, including the visualization pipeline, the data manipulation methods and the various visual metaphors tailored for presenting social networks.

3.1 Pipeline

The visualization process in HiMap follows the general pipeline formalized in [6], as illustrated in Figure 2. It could be divided into two separate parts: the offline data manipulation and the online adaptive visualization. The offline data manipulation includes data collecting, cleaning, and hierarchical clustering stages. It prepares the graph data required for further on-line visualization of the target social networks.

The adaptive visualization part involves the data loading, graph layout, projection and rendering stages. Different from the flat visualization approach that shows all the leaf-nodes of the social network in one view, our HiMap solution clusters them into a hierarchical tree and only presents the nodes within certain depth from the root node of the current view. This strategy greatly reduces the visual clutter commonly found in previous huge graph visualization design. Moreover, through a novel adaptive data loading method, the visual density of each graph view is maintained within the human perception capability. In this way, the data amount required to

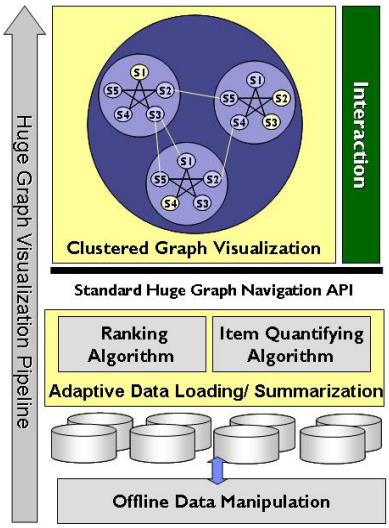


Figure 2: HiMap visualization pipeline.

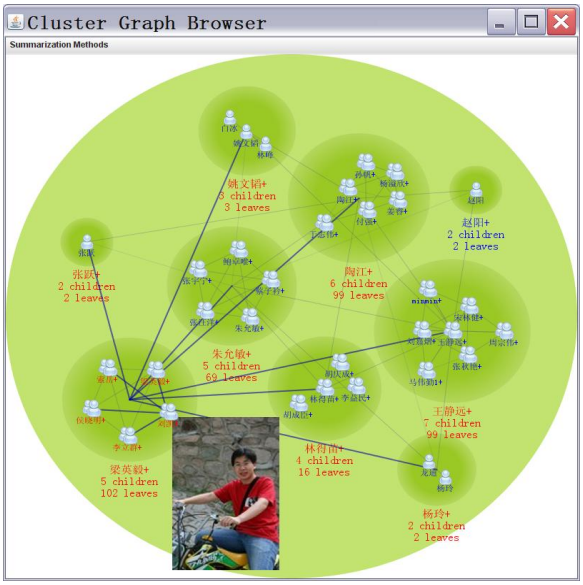


Figure 3: HiMap visualization of an online social network within a university department. Edges between clusters are bundled together by hierarchy. The portrait of the selected people is shown as tooltip and the nodes/edges induced from him are highlighted. Here the induced clusters/edges of its parent cluster are also highlighted to assist the comprehension of hierarchical edge bundling.

load upon each view transition is kept small by the screen's constraint, so that the users could not even find lags except for the smooth animation designed to assist his comprehension. We also maintain a data cache that stores all the view graph data in the user's navigation history. When the user goes back to the previous view, no extra overhead is paid for the second time unless their cache has been swapped out after the timeout.

After the graph data for each view is loaded to memory, layout algorithm is invoked to assign every node a location in the global coordinate system. We currently use SKK-C layout algorithm, a variant of force-directed Kamada-Kawai algorithm for the layout of each hierarchy. It could generate high quality layout result timely for graphs with less than a hundred nodes, quite appealing to our case. The entire layout process works in a bottom-up manner so that the sizes of the clusters are determined before the layout of the graph at their hierarchy. To obtain aesthetically pleasing clustered graph layout and keep visual momentum between consecutive views, the SKK-C algorithm also integrates the hierarchy information and the previous node coordinates to the layout computation.

The projection process maps the coordinate of each node from the global coordinate system to the local coordinate system of the screen for further displaying. Different from the layout process, it works in a top-down manner from the root node of the current view to the deepest viewable hierarchy defined by the system recursively. After all the node's locations are determined, the system starts rendering the nodes and edges according to defined visualization theme.

HiMap also provides a wide suite of interactions to navigate the social network, which leverage the standard huge graph navigation API to probably reload the graph data. They will be detailed in Section 6.

3.2 Data Manipulation

We have visualized data sets from three major sources. One is the IBM internal social network with 0.25 million staffs connected through email, instant message and calendar events extracted by smallblue [23] (Figure 1 only shows a small group of it for legal issues). The second one is the group of 2000+ users from the same department of a university on xiaonei.com [3], a popular online social network in China. Both the user profiles and their relationships are obtained. And the last one is the public DBLP dataset [1], which constructs a co-authorship network of more than 0.6 million researchers in computer science.

Each set of raw data is clustered with methods introduced in [26] which could group the data into binary tree. To obtain balanced hierarchical clustering structure, we invoke them recursively: all the nodes are grouped as the binary tree until some termination condition is triggered (i.e. delta Q value decreases below zero), and the remaining groups becomes the clusters of the first hierarchy; then all these clusters are processed separately to obtain the sub-clusters in the next hierarchy; the algorithm works recursively until the pre-defined maximal tree depth is reached. In fact, we have put a lot effort to generate the balanced hierarchical tree, but the details would be out of the scope of this paper.

The data are stored in the database in mainly two tables: the node table and the edge table. The node table records the affiliated attributes of the node, including the unique id, user name, etc. It also records the hierarchy information, encompassing its parent ID and the depth in the tree structure. To assist the online visualization, we carry out some pre-processings on the graph data. The importance rank of nodes (or clusters) in the same hierarchy is computed according to the ranking algorithms proposed in Section 4.1. This rank determines the sequence to display when the screen size can not accommodate all the nodes in the same hierarchy. Three kinds of node centrality (degree, closeness and betweenness) in the graph are also calculated and recorded to facilitate the node filtering. We pre-compute the hyper-edges between any two clusters in the same hierarchy to reduce the online computation complexity. Every two clusters establish a hyper-edge if there is at least one leaf-edge connecting two nodes in the separate cluster.

3.3 Visual Metaphors

The visual items in HiMap are rendered to facilitate the human perception. The clusters are drawn as circles without explicit frame, background color for each cluster is painted from the center in a descending lightness along the radius to indicate its boundary. The clusters not capable to show its internal structure due to the screen constraints are drawn as a much smaller circle without any sub-cluster (node) in it. To maintain consistency of the entire view, we set the maximal viewable depth to 2, and the sub-clusters inside

each top-hierarchy cluster are drawn as a icon indicating a group of people, tagged with the most representative people's name followed by a "+". When the sub-cluster only represents one leaf node, it is shown as a people icon instead. The background color of the cluster is set according to its depth in the entire tree structure, rendering darker for the deeper depth.

By default, the edges between any two leaf nodes are drawn in the view by straight line. To reduce the visual clutter commonly found in the densely connected graph, we also introduce two edge bundling methods: the geometric edge bundling and the hierarchical edge bundling. The geometric bundling implements the solution similar to [8]. It works by carefully selecting some control points in the graph and forcing all the edges to traverse them, as shown in Figure 1. The other method bundles all the edges between any two upper-hierarchy cluster together and only shows the intra-edges inside each cluster for the lower-hierarchy sub-clusters. An example is given in Figure 3.

HiMap also provides a way to navigate user profile. When the leaf node in the view is moused over, its corresponding profile (including its picture) is shown as tooltip, and all the connecting nodes and induced edges are highlighted to facilitate user's comprehension, as shown in Figure 3.

4 Adaptive Data Loading

This section describes the algorithm to summarize graph to maintain comfortable visual density and readability subject to the changing screen size. It involves two steps (except for the final loading step), first the nodes in each view are ranked by its customized importance. We currently provide two ranking algorithms: the first is a maximal coverage based algorithm seeking to display a minimal set of representative nodes to cover a maximal set of nodes in the same hierarchy, so that the user could find at least one friend of his interested person in the representative set with highest probability. A generalization of this algorithm is also proposed to deal with a broader scope of situation. The second algorithm is to rank the nodes according to the clustered betweenness centrality, so as to display the brokers connecting the clusters first for social network diagnosis purpose. The ranks with the both ranking algorithms are pre-computed and kept in the database, so that the user could switch the view freely as his specified demand with rather low overhead. The second step to summarize the graph is to determine the number of visual items to select in each hierarchy, for which we have designed a recursive allocation algorithm.

4.1 Ranking Algorithms

Ranking algorithm is to determine the selection order of visual items in case the screen size is limited. Without loss of generality, we consider the ranking of the child items of a particular parent cluster C. For the top-hierarchy items, C will be an artificial super cluster of the entire social network graph. The set of the child items of C is denoted as V and the set of edges (including hyper-edges) connecting those in V is denoted as E. The child graph of C is denoted as $G(V,E)$. G is considered as an undirected graph in our case. Given a subset S of V, its coverage set S^+ in V is defined as

$$S^+ = S \cup \{v \in V | \exists e = (v, v^*) \in E, v^* \in S\} \quad (1)$$

Intuitively, to maximize the user's search capability in the view, we need to follow the maximal coverage guideline given below.

Maximal Coverage Guideline: The i highest-rank child items of C should be those that maximize the cardinality (number of members in the set) of their coverage set.

The maximal coverage set problem is NP-hard with no known polynomial time solution. (The classical set covering problem, which is one of the Karp's 21 NP-complete problems [21], is polynomial time Turing-reducible to this maximal coverage set problem.) We introduce the greedy algorithm to approximate it. Here, I_i denotes the set of child items of C with ranks no more than i.

Maximal Coverage Greedy Algorithm: The rank-i child item of C, denoted as v_i, is selected to be the one that maximizes $|I_i^+| - |I_{i-1}^+|$.

The problem with the maximal coverage algorithm is that it does not provide method to break the ties. To handle that, we introduce two minor ranking guidelines followed after the first one.

Given a subset S of V, the coverage strength of item v in set $S^+ - S$ is defined to be the number of edges in E connecting v with items in S.

Maximal Coverage Strength Guideline: The rank-i child item of C should maximize the increase of the total coverage strength of items in $I_i^+ - I_i$.

Maximal Degree Guideline: The rank-i child item of C should maximize its own degree.

With these two minor guidelines, we generalize the ranking algorithm to a maximal weighted-degree algorithm. Denoting S as the selected item set, for any item v in the unselected set $V - S$, we decompose its degree $d(v)$ into $d(v) = d_s(v) + d_c(v) + d_r(v)$, where $d_s(v)$, $d_c(v)$ and $d_r(v)$ are the number of edges in E connecting v to items in set S, $S^+ - S$ and $V - S^+$ respectively.

Maximal Weighted-Degree Greedy Algorithm: Given the selected item set I_{i-1}, the rank-i child item of C is selected to be item v in set $V - I_{i-1}$ with the largest weighted degree $wd(v) = \phi_s d_s(v) + \phi_c d_c(v) + \phi_r d_r(v)$. Here, $\phi_s = d_{max}^{-1}, \phi_c = d_{max}^{-2}, \phi_r = 1$. d_{max} is the degree of the rank-1 item in V.

The maximal weighted-degree greedy algorithm follows the maximal coverage guideline, maximal coverage strength guideline and the maximal degree guideline in a strict priority manner.

The above ranking algorithm is positioned to improve the people search experience of HiMap. While in another important scenario, the users are more interested in finding the brokers that connect different group (cluster) of people. To meet this requirement, we provide another betweenness based ranking algorithm. Particularly, we define a new topology metric for clustered graph, the *clustered betweenness centrality (C_{BC})*. Different from the graph-theory version betweenness centrality, it only considers the shortest path between nodes in separate clusters.

Consider the same scenario of ranking the child items of a parent cluster C. This time, we trace upwards to the parent item of C. In case C is already the super cluster of the entire social network, i.e., these child items are the top-hierarchy items, we do not implement betweenness based ranking, but use the coverage based ranking. Denote the parent item of C as GC, the grand-children item set of GC as V, the set of edges connecting those in V as E and their graph as G(V,E). For each item v in V, its parent cluster is denoted as $p(v)$. The clustered betweenness centrality of item v is defined by

$$C_{BC}(v) = \sum_{s \neq v \neq t \in V, p(s) \neq p(t)} \frac{\sigma_{st}(v)}{\sigma_{st}} \quad (2)$$

where σ_{st} is the number of shortest paths in G from s to t, and $\sigma_{st}(v)$ is the number of shortest paths in G from s to t that pass through v. It is further normalized by dividing through by the number of pairs of items in V with different parent clusters.

Maximal Clustered Betweenness Centrality Algorithm: Given selected item set I_{i-1}, the rank-i child item of C is selected to be item v in the unselected set $V - I_{i-1}$ with the largest C_{BC}.

4.2 Quantifying visual items

The underlying goal to quantify the visual items in HiMap is to display readable social network graph for each possible view subject to the current screen size. Although the readability of a graph is quite a subjective metric, there are still two important factors that indeed

impacts heavily the viewer's mark on graph readability: 1) the number and size of visual items displayed in the screen; 2) the detailed topological manner they are displayed. While the latter factor will be determined by the layout algorithms, which is not changed online in HiMap, we mainly depend on tuning the former factor to control the readability of each graph. In this sense, our method could be thought as controlling the visual density of graphs.

The visual density of graph $G = (V,E)$ shown in screen Γ with size ω (by screen pixels) is defined by

$$VD(G) = \frac{\Upsilon_V \sum_{v \in V} S_v + \Upsilon_E \sum_{e \in e} W_e L_e}{\omega} \quad (3)$$

where S_v, W_e and L_e are the node size of $v \in V$, edge width and edge length of $e \in E$ shown in screen Γ, Υ_V and Υ_E are impacting coefficient of nodes and edges on the visual density. The graph G is said to be *visually dense-free* if $VD(G) \leq VD_{th}$ where VD_{th} is a user-tunable threshold defining his maximal perception capability.

Denote the social network graph under investigation as G, and its top-hierarchy graph as $G_1 = (V_1, E_1)$. Graph $G^* = (V^*, E^*)$ is said to be an *integrated subgraph* of $G = (V,E)$ if $V^* \subseteq V$ and $\forall e = (v_1, v_2) \in E, v_1 \in V^* \wedge v_2 \in V^* \rightarrow e \in E^*$.

Recursive Item Quantifying Algorithm: Given the social network graph G, the algorithm works in three steps: 1) Find the maximal visually dense-free integrated subgraph G_1^* of G_1 subject to the current screen size ω. The algorithm exits here if $G_1^* \neq G_1$; 2) Calculate the extra screen size $\overline{\omega}$ after feeding the graph G_1. Proportionally assign $\overline{\omega}$ to all the clusters $v \in G_1$, according to the number of leaf nodes inside each v; 3) Quantify the items selected inside each $v \in G_1$ by recursively invoking this algorithm subject to the screen size assigned to it.

Upon the basic quantifying algorithm, we introduce several control parameters according to HiMap visualization guideline.

Maximal Visualization Depth (d_{max}): It defines the maximal depth of items selected to show in each view from the current super cluster. When the recursive quantifying algorithm runs into such depth, it simply exits after Step 1. In HiMap, we set this value to 2.

Minimal Visualization Depth (d_{min}): It defines the minimal depth of leaf items in each view from the current super cluster. Each item with depth smaller than that should have at least one child item selected if there is any. In HiMap, we set this value to 2.

Maximal Visualization Breadth (b_{max}): It defines the maximal number of items to expand in Step 3 of the quantifying algorithm. Showing the hierarchies of too many clusters will degrade the user's capability to comprehend the graph, we currently set this value to the magic number of seven plus two [24].

Overview Depth ($d_{overview}$): Complemental to the maximal visualization breadth, another common demand is to give an overview to the entire social network graph. We therefore define an overview depth parameter that the items with depth no more than that are always selected. We currently set this depth to 1. This requirement will override the above three constraints.

The pseudocode to describe the integrated algorithm of item quantifying and subsequent data loading is given below.

5 Clustered Graph Layout

HiMap layout algorithm is designed based on the Kamada and Kawai method [20], which is a force-directed algorithm that minimizes the energy of a simulated physical spring systems, given in (4). The desirable geometric distance between any two nodes is selected to be the graph shortest path between them.

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \omega_{ij} (\| X_i - X_j \| - d_{ij})^2 \quad (4)$$

where X_i is the layout position of node i, d_{ij} is the desirable distance between node i and j, and ω_{ij} is spring strength with $\omega_{ij} = d_{ij}^{-2}$.

The major improvements of HiMap layout algorithm upon the above basic approach lie in two aspects: 1) it generalizes and optimizes the Kamada-Kawai algorithm to work recursively for the clustered graph; 2) it introduces an adaptive stabilizing term to maintain visual momentum between consecutive graph layouts. We therefore name our layout algorithm as Stable Kamada-Kawai layout algorithm for Clustered graph (SKK-C). Below we describe the SKK-C algorithm in detail.

SKK-C Algorithm: Given a clustered graph $G = (V,E)$ and denoting its top-hierarchy graph by $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ ($n = |\widetilde{V}|$), the process to layout G includes three steps: 1) Invoke this algorithm recursively to layout the child graph of any non-leaf cluster $v_i \in \widetilde{V}$; 2) Calculate the boundary for each such cluster v_i; 3) Calculate the layout of $\widetilde{G}$ by minimizing the summed energy

$$(1-\alpha) \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \omega_{ij} (\| X_i - X_j \| - d_{ij})^2 + \alpha \sum_{i=1}^{n} \mu_i \| X_i - \lambda X_i^{'} \|^2 \quad (5)$$

where X_i and $X_i^{'}$ are the positions of cluster (node) v_i in the current and previous view respectively. $\lambda > 0$ is a scaling factor, d_{ij} is the desirable distance between v_i and v_j, ω_{ij} is the strength of spring connecting v_i and v_j, μ_i is the strength connecting v_i with its previous location, $\alpha \in [0,1]$ is the factor trading off the layout term and the stabilizing term of the energy equation.

Algorithm 1 SUMMARIZENODE(R, d, ω, ρ)

```
/* R: root node, d: summarize depth, ω: window size, ρ: desired
visual density, ω_min: minimal window size for a cluster */

/* Initialization and find the maximal dense-free graph */
σ ← 0, r[i] ← 0, k ← 0, ExpandNs[k] ← NULL
ChildG ← CHILDGRAPH(R)
ChildNs ← CHILDNODES(R)                    // listed by rank;
SubNs ← DENSEFREENODES(ChildG, ω, ρ)       // listed by rank;
if Len(SubNs) == 0&&Len(ChildNs) > 0&&d <= d_min then
    SubNs[0] ← ChildNs[0], Len(SubNs) ← 1

/* Load nodes */
for i = 0 to Len(SubNs) − 1 do
    Load SubNs[i];
    if !IsLeaf(SubNs[i])&&(k < b_max||d < d_min) then
        ExpandNs[k] ← SubNs[i]
        r[k] ← GETLEAFRATIO(SubNs[i])
        σ ← σ + r[k], k ← k + 1
if d <= d_overview then
    for i = Len(SubNs) to Len(ChildNs) − 1 do
        Load ChildNs[i]

/* Compute extra window size */
ρ̂ ← COMPUTEVISUALDENSITY(SubNs, ω)
ω ← ω × (1 − ρ/ρ̂)

/* Expand hierarchies inside the nodes */
if d < d_max then
    for i = k − 1 to 0 do
        if ω < ω_min&&d >= d_min then
            break;
        ω̂ ← ω×r[i]/σ
        if ω̂ >= ω_min then
            v ← SUMMARIZENODE(ExpandNs[i], d + 1, ω̂, ρ)
            ω ← ω − ω̂ + v
        else if d < d_min then
            SUMMARIZENODE(ExpandNs[i], d + 1, ω_min, ρ)
        σ ← σ − r[i]
return ω
```

The major difference of the layout term for the clustered graph is on the determination of d_{ij} and ω_{ij}. The major consideration is to avoid the overlapping of clusters while preserving the high-quality layout aesthetics. We set

$$d_{ij} = 2\bar{r} \times g_{ij} + \theta \times \frac{r_i + r_j}{2},\ \omega_{ij} = 1 \quad (6)$$

where r_i is the diameter of the circular bound of cluster v_i, $\bar{r}$ is the average diameter of all the clusters in $\widetilde{V}$, g_{ij} is the graph shortest path between v_i and v_j, θ is the scaling parameter.

The main idea here is to scale the desirable distance between clusters by a factor of the average cluster radius and then append it with the sum of the two cluster's radius. After these operations, the desirable distance between any two clusters still remain proportional to their graph shortest path in case $\theta = 1$. The advantage over the typical value of d_{ij} is that the clusters will have less chance to stay overlapped with each other after the layout. The key observation here is that we need not to maintain the equivalence of distance unit in different hierarchies. The scaling parameter θ is also provided to reduce the cluster overlapping as it is set larger than 1. Besides, ω_{ij} is set to 1 to put more weight on the pair of clusters with longer distance, compared with the previous value of $\omega_{ij} = d_{ij}^{-2}$. This approach also helps to prevent the cluster overlapping.

The main idea of the stabilizing term [7] of Equation (5) is to add additional springs with zero initial length to drag each node towards its old location if it is present in the previous view. In our implementation, we integrate this term into the CG Solver [13], which is one of the effective method to compute the energy minimization point for Kamada-Kawai layout algorithm. It works in an iterative manner, and in each iteration, it computes a new location for each node in the layout set from the location of last iteration. When the termination condition is met such that there is not a single node moving significantly enough from the last position, the iteration will stop and all the node locations are finalized.

6 Interaction and Animation

HiMap is equipped with a variety of interactions to allow the user to navigate the social network, from a top view to every details. Here we will highlight the two specific zooming operations designed for clustered graph, and the animations customized to keep user's visual momentum. At last, we will briefly introduce our interactive user interface.

6.1 Zooming Operations

Semantic Zoom-in/Zoom-out: Different from the normal geometric zooming operation, the semantic zooming will bring new visual items to the view upon magnifying and retract some old visual items with low importance ranking upon minifying. This method is built upon our adaptive visualization technique. For a previous screen Γ with the view of the child graph of super cluster C, by the adaptive visualization method given in Section 4, we will select to show the child and grand-child items of C subject to the current screen size. Upon semantic zoom-in, a virtual zoom-in window Γ' is chosen according to the current cursor position and the zoom granularity, as shown in Figure 4(a). The zoom granularity is set to 60% by default and can be changed by users. After the zoom-in, only the child items of C in Γ' is selected, and the items inside them are picked up again by the adaptive loading algorithm, so that more details could be visualized. Figure 4(c) shows an example for the semantic zooming.

Hierarchical Drill-in/Roll-up: Like the semantic zooming, the hierarchical drill-in also reveals new contents and requires data reloading after the operation. The major difference is that the semantic zooming will retain in the previous hierarchy while the drill-in/roll-up allow the users to navigate through different hierarchies. An example is given in Figure 4(d).

6.2 Animations

HiMap present animations during the operation generating view changes. The major objectives for the animation are: 1) Keep the user's mental map on the social network graph; 2) Improve the user's capability to learn new contents.

For the first objective, we have designed the SKK-C layout algorithm to stabilize the consecutive views. And for the second objective, we customize the animations of different operations to stages of motions so that the user could focus on different types of changes at each stage. The detailed animations are described below (reverse animation is omitted), and the series of charts illustrating the typical drill-in animation are given in Figure 5 as an example.

Semantic Zoom-in: Stage I: The top-hierarchy clusters remaining in the new view will smoothly move to their new locations, with their shapes morphing to the new shape, while the top-hierarchy clusters absent in the new view will progressively move outside the screen as if the entire view expands to squeeze them out. *Stage II:* The new child items of the top-hierarchy clusters shown only after the zoom-in operation grow from the center of its parent cluster, like the blooming.

Hierarchical Drill-in: Stage I: The child items of the drilled-in cluster will smoothly move to their new locations, with their shapes morphing to the new shape. Note that by our summarization and ranking methods, all these child items will be visualized in the new view to preserve user's momentum. *Stage II:* The child items of the drilled-in cluster shown only after the drill-in operation fade in to the new view. *Stage III:*The child items of the top-hierarchy clusters of the new view grow from the center of its parent cluster.

6.3 Control Interface

Figure 6 visualizes the DBLP dataset, indicating the computer science academic community with more than 0.6 million researchers. At the right of the main panel, we create a column of control interface that allows the users to adjust and tune the initial view of the social network. Users could choose summarization methods to show different types of role players in the network, and search with researcher names to go directly to the right community (cluster) they are interested. People icon configurations allow the user to switch between virtual icon and real picture and also specify the icon size to allow more people in one view. The parameters introduced in Section 4.2, such as graph visual density, maximal/minimal visualization depth, maximal visualization breadth (Max Cluster Number) and overview depth, are also tunable through the user interface.

7 Evaluation

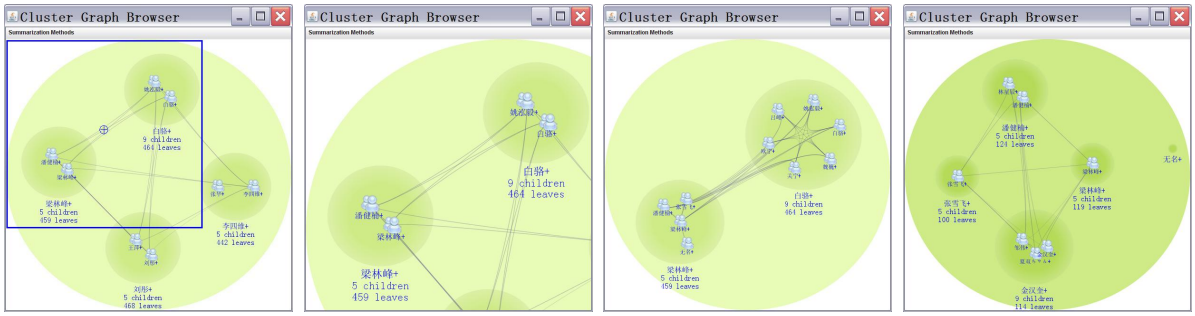
This section evaluates the performance of the adaptive data loading techniques and SKK-C layout algorithm in HiMap.

7.1 Setup

All the experiments here are carried out with our demo implementations using the first data set describing the smallblue network. In each experiment slot, we configure the demo to automatically navigate the graph view inside every cluster within a maximum depth of 2 from the root super cluster, adding up to 117 different views. The navigation follows the standard preorder tree traversal so that stable layout algorithm could take effect by knowing the previous higher-hierarchy layout. By default, the maximal and minimal visualization depth are set to 2, the maximal visualization breadth is set to a larger number of 20 to enable more complicated situations.

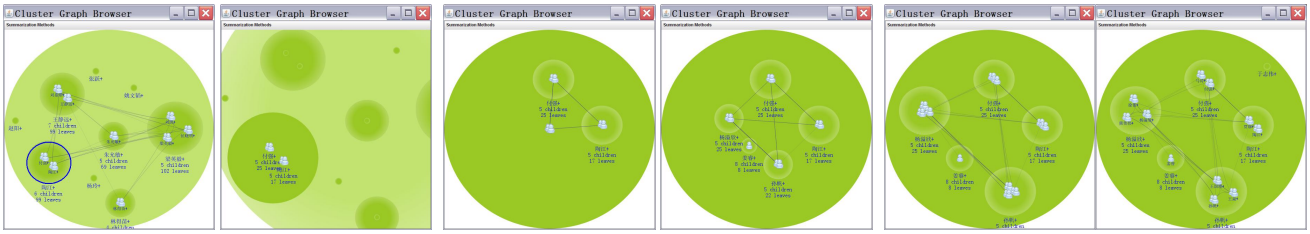
7.2 Graph Summarization

In this part, we compare the two types of ranking-based adaptive graph summarization algorithms with the metric-based filtering algorithms. By the ranking-based algorithms, the number of visualized items will be selected to according to the desired visual density. While by the filtering algorithms, it is only determined by



(a) (b) (c) (d)

Figure 4: Zooming operation in HiMap: (a) Initial view, the blue frame shows the zoom-in window Γ'; (b) After geometric zoom-in, the items are only signified; (c) After semantic zoom-in, more items are visualized adaptively; (d) After drilling in the cluster in the left of sub-figure (a).



(a) Stage I (b) Stage II (c) Stage III

Figure 5: Drill-in animation in HiMap, the circular blue frame in the leftmost figure indicates the focused cluster.

the graph characteristics by providing a lower bound for a predefined metric. All the items with metric higher than that is visualized. We have evaluated three of this kind of algorithms using degree/closeness/betweenness centrality as the metric respectively. (The metric values are linearly normalized to [0,1].)

We plot in Figure 7 the average of actual visual density (as defined in Eq. (3)) of some selected views during the graph navigation. By ranking based algorithms, we set the desired visual density to $0 \sim 0.2$, as in the bottom X axes. By metric based algorithms, we set the metric lower bound to $1 \sim 0$, as in the top X axes. To better reveal the performance of ranking based algorithms, we only select the views that are not fully summarized at the desired visual density of 0.2. Note that, this does not affect the performance of metric based algorithms. As shown by Figure 7, the ranking algorithms achieve almost their desired visual density, with a maximal error below 10%. It shows their capability in accurately controlling the graph visual density. On the other hand, the metric based algorithms will find it hard to operate the graph density smoothly and linearly via tuning the metric lower bound.

Figure 8 illustrates the average of coverage percentage of the selected items' coverage set over all the items within any given view. As shown by the figure, the average coverage percentage reaches 60% if only 20% items in each view are selected using the maximal coverage ranking algorithm. Meanwhile, the centrality metric based algorithms could only reach 48%, 47% and 32% coverage percentage respectively. The ranking algorithm shows a consistent gain of more than 20% before the selected item percentage increases above 30%.

7.3 Layout Algorithm

We record the numbers of overlapped cluster pairs in all the views during the navigation and translate them into the probability of cluster overlaps given the number of visualized clusters in each view. As shown in Figure 9, with the original Kamada-Kawai layout algorithm, the average cluster overlap probability reaches 40% as the number of cluster per graph view increases above 15.

Given in Figure 9, it is shown that, after adding the cluster radius into the ideal distance between each cluster pair ($d_{ij} = g_{ij} + \frac{r_i + r_j}{2}$), the overlap probability decreases dramatically to about 10% even with more than 15 clusters in each view. Introducing the distance scaling ($d_{ij} = 2\bar{r} \times g_{ij} + \frac{r_i + r_j}{2}$), the overlap probability is further halved. Finally after we set $d_{ij} = 2\bar{r} \times g_{ij} + \theta \times \frac{r_i + r_j}{2}$ and tune θ to be $1 + [\frac{n-10}{10}]^+$, the SKK-C algorithm will limit the overlap probability below 1.6% for all the evaluated situations, without greatly affecting the layout aesthetics.

8 Conclusion

This paper presents HiMap, a visualization system that displays the hierarchical structure and relationships of social networks. HiMap puts the first emphasis on eliminating the visual clutter naturally raised by the huge user base of state-of-the-art commercial online social networks. The visual density based adaptive data loading technique and the optimized layout algorithm for clustered graph are both designed, implemented and evaluated for this purpose. Our evaluation results demonstrate that HiMap is capable of rigidly controlling the visual density of graph view, while limiting the cluster overlap probability to rather low level.

Acknowledgements

We thank Xiaoxiao Lian, Xiaohua Sun, Weijia Cai and Hong Ren for their valuable advices on the visual interface design and the anonymous reviewers of PacificVis'09 for their insight comments.

References

[1] The dblp computer science bibliography, http://www.informatik.uni-trier.de/ ley/db/.

[2] Touchgraph, http://www.touchgraph.com.

[3] xiaonei.com, http://www.xiaonei.com.

[4] Alexa Internet Inc. *Alexa Top 500 sites, http://www.alexa.com*, September 2008.

[5] D. Auber, Y. Chiricota, F. Jourdan, and G. Melancon. "Multiscale Visualization of Small World Networks". In *IEEE Symposium on Information Visualization*, 2003.

[6] G. D. Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1998.

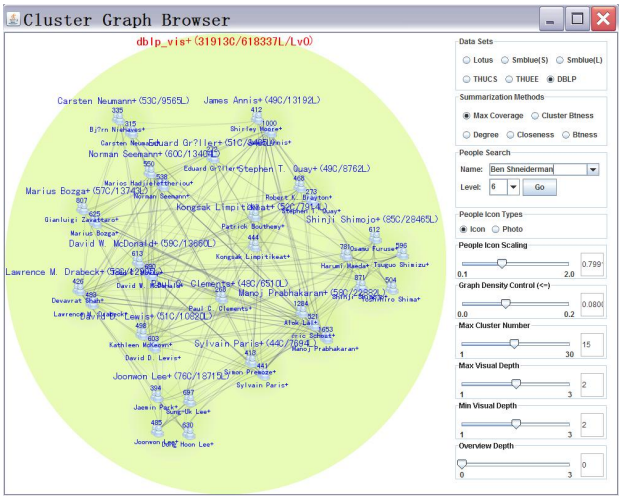


Figure 6: HiMap visualization of DBLP dataset.

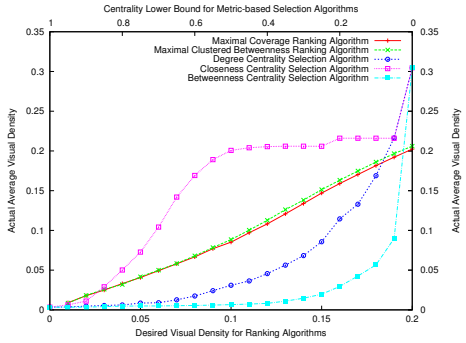


Figure 7: Average visual density of graph views during the navigation.

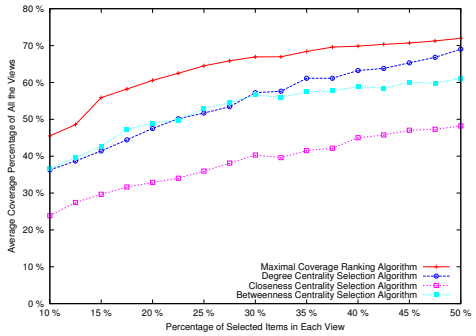


Figure 8: Coverage performance of summarization algorithms.

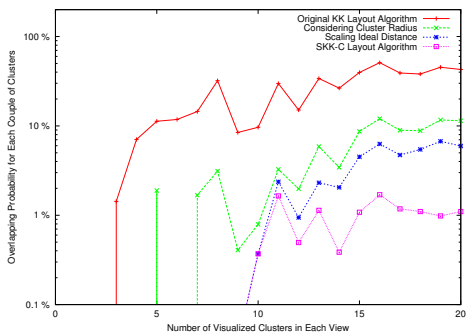


Figure 9: Cluster overlap probability with different layout algorithms.

[7] N. Cao, S. Liu, L. Tan, and M. X. Zhou. "Context-Preserving Dynamic Graph Visualization". In *IEEE Symposium on Information Visualization, poster paper*, 2008.

[8] W. Cui, H. Zhou, H. Qu, P. C. Wong, and X. Li. "Geometry-Based Edge Clustering for Graph Visualization". In *IEEE Symposium on Information Visualization*, 2008.

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

[10] P. Eades and Q.-W. Feng. "Multilevel Visualization of Clustered Graphs". In *Proceedings of the International Symposium on Graph Drawing*, London, UK, 1997. Springer-Verlag.

[11] Q. Feng. *Algorithms for Drawing Clustered Graphs*. PhD thesis, University of Newcastle, 1997.

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

[13] E. R. Gansner, Y. Koren, and S. North. "Graph drawing by stress majorization". In *International Symposium on Graph Drawing*, 2004.

[14] J. Heer and D. Boyd. "Vizster: Visualizing Online Social Networks". In *IEEE Symposium on Information Visualization*, 2005.

[15] N. Henry and J.-D. Fekete. "MatrixExplorer: a Dual-Representation System to Explore Social Networks". In *IEEE Symposium on Information Visualization*, 2006.

[16] N. Henry, J.-D. Fekete, and M. J. McGuffin. "NodeTrix: a Hybrid Visualization of Social Networks". In *IEEE Symposium on Information Visualization*, 2007.

[17] I. Herman, G. Melancon, and M. S. Marshall. "Graph Visualization and Navigation in Information Visualization: A Survey". *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, 2000.

[18] M. L. Huang and P. Eades. "A Fully Animated Interactive System for Clustering and Navigating Huge Graphs". In *Proceedings of the International Symposium on Graph Drawing*, 1998.

[19] Y. Jia, J. Hoberock, M. Garland, and J. C. Hart. "On the Visualization of Social and other Scale-Free Networks". In *IEEE Symposium on Information Visualization*, 2008.

[20] T. Kamada and S. Kawai. "An algorithm for drawing general undirected graphs". *Inf. Process. Lett.*, 31(1):7–15, 1989.

[21] R. M. Karp. "Reducibility among combinatorial problems". In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[22] J. Leskovec and C. Faloutsos. "Sampling from large graphs". In *ACM SIGKDD*, 2006.

[23] C.-Y. Lin, K. Ehrlich, V. Griffiths-Fisher, and C. Desforges. "SmallBlue: People Mining for Expertise Search". *IEEE MultiMedia*, 15(1):78–84, 2008.

[24] G. A. Miller. "The Magical Number Seven, Plus or Minus Two". *The Psychological Review*, 63(2):81–97, 1956.

[25] K. Misue, P. Eades, W. Lai, and K. Sugiyama. "Layout Adjustment and the Mental Map". *Journal of Visual Languages & Computing*, 6(2):183–210, June 1995.

[26] M. E. J. Newman. "Fast algorithm for detecting community structure in networks". *Physical Review E*, 69:066133, 2004.

[27] A. Perer and B. Shneiderman. "Balancing Systematic and Flexible Exploration of Social Networks". *IEEE Transactions on Visualization and Computer Graphics*, 12(5):693–700, 2006.

[28] D. Rafiei and S. Curial. "Effectively Visualizing Large Networks Through Sampling". *IEEE Visualization Conference*, 2005.

[29] Z. Shen, K.-L. Ma, and T. Eliassi-Rad. "Visual Analysis of Large Heterogeneous Social Networks by Semantic and Structural Abstraction". *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1427–1439, 2006.

[30] J. M. Six and I. G. Tollis. "Effective Graph Visualization Via Node Grouping". In *IEEE Symposium on Information Visualization*, 2001.

[31] F. van Ham and J. J. van Wijk. "Interactive Visualization of Small World Graphs". In *IEEE Symposium on Information Visualization*, pages 199–206, 2004.

[32] F. van Ham and M. Wattenberg. "Centrality Based Visualization of Small World Graphs". *Eurographics*, 27(3), 2008.

[33] Wikipedia. *List of Social Networking Websites, http://en.wikipedia.org/wiki/List_of_social_networking_websites*.