Hierarchical Focus+Context Heterogeneous Network Visualization

Lei Shi* Qi Liao† Hanghang Tong‡ Yifan Hu§ Yue Zhao¶ Chuang Lin‖

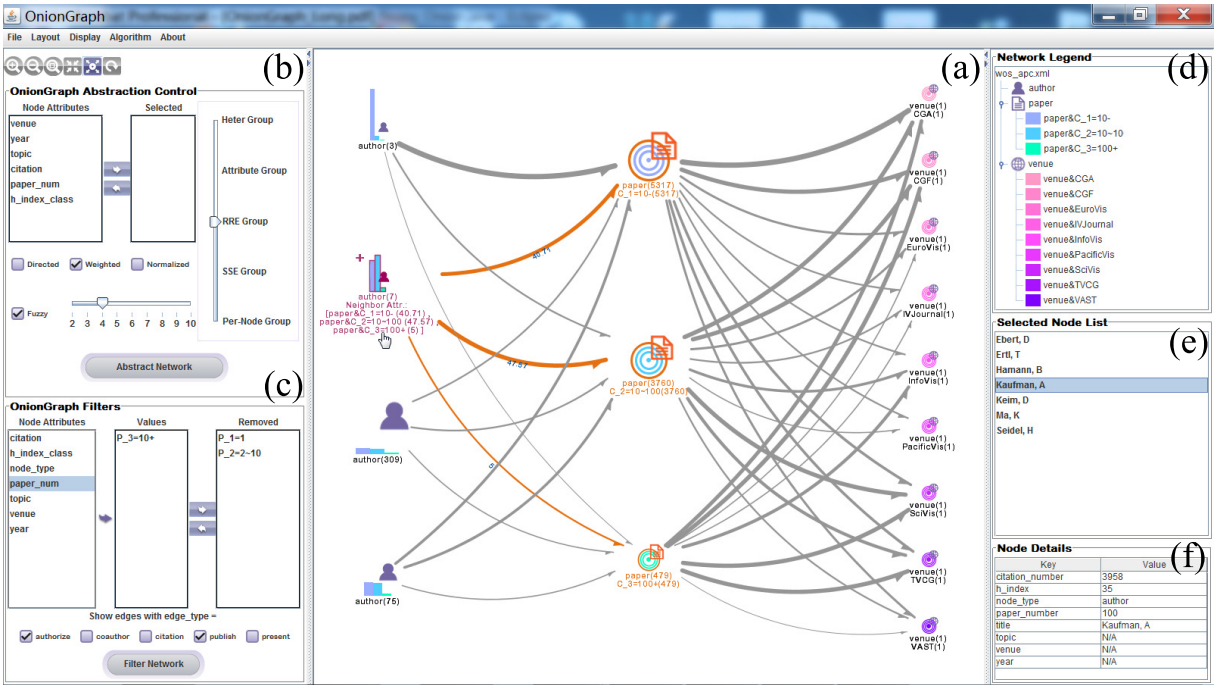


Figure 1: OnionGraph interface showing the bibliographic network in the visualization community. (a) Main OnionGraph panel visualizing the author-paper-venue heterogeneous network. Venues (right column) are expanded into single journal/conference entities; papers (central column) are expanded by their citation count groups; authors (left column) are expanded by their neighborhood attributes, i.e., the publication profile of low/medium/high-citation papers. The layout improves PivotGraph grid-like layout [27]. (b) Configuration panel for OnionGraph abstraction. The current abstraction control specifies the profile of the selected author node in the left column of the main panel. (c) Filter panel which is configured to only show authors who publish more than 10 papers, and edges which connect author-paper ("authorize" type) and paper-venue ("publish" type). (d) Legend panel showing the icons and colors used in the current network. (e) The selected node list which includes authors with many high-citation papers. (f) Details of "Kaufman, A", all the statistics (e.g., h-index) are computed within the visualization paper dataset.

Abstract

Aggregation is a scalable strategy for dealing with large network data. Existing network visualizations have allowed nodes to be aggregated based on node attributes or network topology, each of which has its own advantages. However, very few previous systems have the capability to enjoy the best of both worlds. This paper presents OnionGraph, an integrated framework for exploratory visual analysis of large heterogeneous networks. OnionGraph allows nodes to be aggregated based on either node attributes, topology, or a mixture of both. Subsets of nodes can be flexibly split and merged under the hierarchical focus+context interaction model, supporting sophisticated analysis of the network data. Node aggregations that contain subsets of nodes are displayed with multiple concentric circles, or the onion metaphor, indicating how many levels of abstraction they contain. We have evaluated the OnionGraph tool in two real-world cases. Performance experiments demonstrate that on a commodity desktop, OnionGraph can scale to million-node networks while preserving the interactivity for analysis.

1 Introduction

Information networks are intensively studied nowadays and many of them are *heterogeneous* in that their nodes and edges are of different types. Each type can be associated with a few attributes.[1] For example, in a bibliographic information network, a node can be an author with affiliation, a paper with topic, or a venue (i.e., conference/journal) with location. An edge can represent the relationship of citation, authorization, presentation, etc. With this semantic augmentation, analyzing a heterogenous network can lead to more insights than its homogeneous counterpart. Besides knowing the authors in the center of a co-authorship network, it is also possible to detect the authors with highly cited papers at a prestigious

*State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences. Email: shil@ios.ac.cn

†Department of Computer Science, Central Michigan University. Email: qi.liao@cmich.edu

‡Computer Science Department, City College, CUNY. Email: tong@cs.ccny.cuny.edu

§AT&T Labs Research. Email: yifanhu@research.att.com

¶Tsinghua University, Email: zhao-yue11@mails.tsinghua.edu.cn

‖Tsinghua University. Email: chlin@tsinghua.edu.cn

[1] Heterogeneous networks are sometimes referred to as multivariate or multifaceted networks.

conference.

The problem considered here is how to visualize a large heterogeneous network in a way that allows a user to perceive and explore both topology patterns and node/edge attributes attached on the network. Similar to many visualization methods on large homogeneous networks [7] [4] [5], we focus on the top-down approach that presents the network with an initial overview and allows drilling down to details through interactions. Ultimately, we consider two problems. First, the **summarization** problem: how to create the visual abstraction of a large heterogeneous network with both topology and attribute information. For example, the multi-scale visualization by hierarchical graph clustering [7] is one popular approach when only topology information is considered. Second, the **navigation** problem: which interaction model to apply to guide the user from the initial visual abstraction to analyzing the low-level heterogeneous network patterns. For example, in the multi-scale network visualization, hierarchy-traversing is a typical method for exploring a large network structure.

Despite a wealth of literature in the network visualization research [7] [14] [4], only a few are designed for heterogeneous networks. PivotGraph [27] and OntoVis [21] are early works addressing such needs, but none of them focus on the visual exploration of a large heterogeneous network. In fact, achieving such a goal is nontrivial. First, previous methods to cluster the network for visualization are based on either topology or attribute information, but not both. Second, the visual summary based on clustering should present interpretable results in that there is a clear meaning for each cluster. For example, by topology-based graph clusterings, each cluster indicates a group of nodes with denser internal connections than external ones. Nevertheless on the heterogeneous network, a straightforward method to cluster a dense group may be inappropriate because dense groups can have rather diversified internal attribute distributions. Third, traditional clusterings on single-source information, either topology or attribute, generate self-contained local structures, making the hierarchy-traversing interactions valid. Nevertheless, in navigating heterogeneous networks, the cross-cluster connections are sometimes more important than the internal ones, in which case the hierarchy-traversing interaction model may not be the best choice.

In this paper, we present OnionGraph, an integrated framework for exploratory visual analysis of large heterogeneous networks [2]. OnionGraph creates five hierarchies on the network, from the top/coarsest-level heterogeneous abstraction to the bottom/finest-level per-node granularity, as shown in the top-left hierarchy control panel of Figure 1. The main OnionGraph view, as in the center of Figure 1, depicts a network by OnionGraph abstraction after a few navigation operations. Each node group in the view is associated with a separate or shared abstraction profile customized by user. For example, the selected node group in Figure 1 (in dark-red) has the same abstraction setting as the other three author groups in the same column, after an expansion operation from their shared parent node. The OnionGraph abstraction can be further simplified by adding node/edge attribute filters, so that the interesting part of the network is kept and enlarged for detailed analysis. A list of selected network nodes is shown in the center-right part of the OnionGraph interface, along with a content panel underneath, displaying details of a particular node.

This paper has three major contributions. First, we propose a suite of clustering algorithms organized in a top-down manner to generate hierarchies over the heterogeneous network. The algorithms explicitly combine the topology and attribute information while guaranteeing a finer granularity as the user drills down to a lower hierarchy. Second, we introduce the "onion" visual metaphor to represent the basic node aggregation in the resulting heterogeneous network abstraction. Hierarchy information is revealed by drawing a certain number of circles on the node. Modified grid-based and force-directed layout algorithms are proposed to map the OnionGraph to the screen space. Third, we develop the hierarchical focus+context interaction model in navigating OnionGraph abstractions. Users can determine the abstraction setting on each group of nodes to generate a fully customized heterogeneous network visualization, which can potentially reveal novel patterns. A typical interaction trail contains no more than a few double-clicks.

The rest of this paper is organized as follows. Section 2 summarizes the related work. Section 3 gives an overview of the OnionGraph framework and the hierarchical focus+context interaction model. Details on the algorithms and the visualization design are given in Section 4 and Section 5 respectively. Section 6 presents two case studies with OnionGraph. Finally, Section 7 concludes the paper.

2 Related Work

The literature on heterogeneous network visualization can be roughly classified into relationship visualization and attribute visualization. The former generally inherits the node-link graph metaphor, while the latter is mostly built on statistical charts over selected attributes.

2.1 Relationship Visualization

Wattenberg pioneered PivotGraph [27], which is an attribute-centric node-link visualization of heterogeneous networks. PivotGraph leverages a roll-up operation to pivot the nodes with the same value on one or two attributes into aggregations. The attributes used can be picked manually to generate different PivotGraph views. In a data selection operation, the network can be reduced to only show node aggregations with specified attribute values. Combining the roll-up and selection operations, PivotGraph supports powerful attribute-centric analysis over networks. OnionGraph shares the similar idea in the first two levels of abstraction on node attributes. Beyond the static attribute selection in PivotGraph, OnionGraph allows dynamic aggregation of different portions of the network by separate attributes, which provides more flexibility in solving exploratory network analysis tasks. On lower-level abstractions, OnionGraph's motivation is fundamentally different. Rather than aggregating the network nodes solely by attribute values, OnionGraph combines the attribute and topology information.

In OntoVis [21], Shen et al. proposed the method of semantic and structural abstraction based on the ontology graph of heterogeneous social networks. On attribute analysis, network is filtered by selected nodes in the ontology graph. On structural abstraction, OntoVis provides methods such as degree-one node and duplicate path reductions. OnionGraph is similar to OntoVis in that it also considers both the network semantics and topology. However, the goal is different. OntoVis focuses on visually pruning a large heterogeneous network into a smaller and simpler abstraction for static visualization, while OnionGraph extends to support navigation from the top-level abstraction.

There are several other works creating the static overview of heterogeneous networks, but none of them allow hierarchical visual exploration, which becomes essential when the network size increases significantly. Semantic Substrate [23] proposed a user-defined layout method to place nodes in non-overlapping regions according to their attributes. GraphDice [8] applied a scatterplot visual metaphor to the overview of multivariate networks. FacetAtlas [10] extracted the multifaceted entities and relationships from a collection of documents. By applying a density map based visual metaphor, both global and local connection patterns are revealed

[2] There is another Onion notation in graph drawing literature [24]. However, both the visual metaphor and the usage are quite different. This previous notation is designed for tree-like hierarchical graphs and applied mostly in software visualization such as drawing UML class diagrams.

for analyzing the rich text corpora. Each entity in FacetAtlas is encoded by one visual node explicitly, while OnionGraph introduces node aggregation which is more suitable for large networks.

2.2 Attribute Visualization

Attribute visualization is an important subject in network visualization tasks [17], e.g. finding the node/link with certain attribute value. These tasks involve detailed inspection of network attributes, which demands a design significantly different from traditional node-link visualizations. NetLens [15] is an innovative interface built for such needs. Based on a content-actor data model, NetLens creates a series of statistical charts (e.g. bar charts) upon attribute queries. The content and actor views are shown side-by-side, allowing filters and data flows within and between them. Complex queries are interactively customized to meet the user's analysis requirements. FacetLens [18] exhibits a similar design, but introduces a linear facet to enable navigation and comparison on ordinal dimensions (e.g. time). FacetLens also allows a pivot operation to drill down to node details after various filters are applied.

Due to the complexity to query network attributes, there is a need to manage, retrieve and traverse user's analysis history. GraphTrail [11] is a visualization system designed for such a goal. By linking sequential network attribute views into a trail, GraphTrail enables a user to surf within the analysis process. The basic mechanism only requires simple drag-and-drop operations.

2.3 Interaction Methods

On manipulating network hierarchies, typical interactions include the hierarchy navigation and editing. In [12], Elmqvist and Fekete classified the hierarchical aggregation based visualization into five types: above traversal, below traversal, level traversal, range traversal and unbalanced traversal. The navigation methods generally work to change the hierarchy setting within each type of the classification or switch between two different types. In [7], Auber et al. proposed the method to start from an above traversal and leverage an overview+detail navigation to create a below/range traversal view on the focus. ASK-GraphView [4] allows the user to click on each node aggregation to expand with any traversal type and generates an unbalanced traversal. Similarly, Topological Fisheye [13] enables an interactive switching among unbalanced traversals by specifying some focuses on the network. GrouseFlock [6] provides high-level hierarchy modification operators based on the low-level delete and merge operations.

The focus+context interaction is a classical technique for network visualizations. The hyperbolic visualizers [16] allow a user to focus on some details while preserving the context of the entire network. Topological Fisheye [13] achieves the similar level-of-detail rendering by a pre-computed multi-level coarsening tree. In [26], Van Ham and Perer proposed a method to start the network analysis from a search, where the focus is essentially the search result. Network context is expanded by a Degree-of-Interest diffusion from the initial focuses.

3 OnionGraph Overview

3.1 Principle

Stratified Semantic+Topological Abstraction. As mentioned before, neither the attribute-based nor the topology-based network visualization method alone can well serve the exploratory analysis tasks such as "Is there any VAST paper heavily cited by both TVCG and CGF papers?". Moreover, a flat combination of these two generates complex and fragmented network abstractions. For example, partitioning a social network according to both the user's community and his profile leads to many tiny clusters. In OnionGraph, we introduced a stratified semantic+topological principle. In high-levels, the large network is aggregated by a combination of semantic information (node type and attributes). Interesting portions of

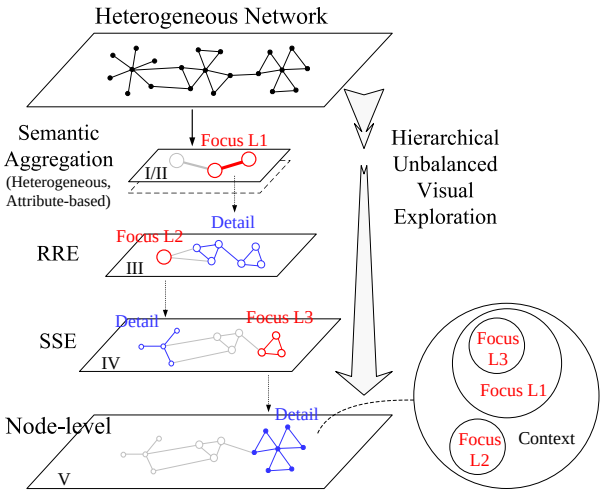


Figure 2: OnionGraph structure featuring five hierarchies below the original network: networks by semantic aggregations (SA) on node type (heterogeneous abstraction) and node attributes, Relative Regular Equivalence (RRE), Strong Structural Equivalence (SSE), and the node-level network in the finest granularity. In each hierarchy, the network can be expanded on certain focuses (red regions) into their lower-hierarchy details (blue regions).

the abstraction can be drilled down in-situ by exploiting topological features. After steps of user navigation, a stratified network view having different levels of abstraction is created on demand to serve the complicated heterogeneous network analysis tasks. Sketched examples are given in Figure 2.

The primary goal of the stratified framework is to achieve the level-of-detail viewing: each lower-level hierarchy should present significantly richer network complexity than its parent hierarchy. This accounts for why semantic aggregation is placed on top of topological methods, and also the design of the 5-level structure (Section 3.2). With appropriate node attribute selected and an optional binning operation, semantic aggregation can always create a neat abstraction of the entire network. On the other hand, most real networks have limited topology redundancy. Compressing a large network purely by topology methods (e.g. structural equivalence) can lead to another cluttered network. Also, observations on real network data show that the topologically identical network nodes can have very high probability to possess the same node attributes, while the opposite is in most cases not true: the identity on node attributes has little to do with their topology positions.

Unbalanced Exploration. Most current hierarchical network visualization methods assume a pre-computed hierarchy. Users can only follow existing trails to explore the network and discover patterns. This places many limitations on the visual analytics capability. Moreover, the network hierarchies, such as those by graph clusterings, are often sensitive to the parameter applied. Users can hardly understand why or why not some parts of the network are grouped together.

In contrast, OnionGraph features an unbalanced visual exploration design on heterogeneous networks, as shown by the trail in Figure 2. First, we apply a few well-defined network abstraction algorithms, each resulting in an unambiguous partition of the network. Users are guided by algorithm heuristics, so that they can understand the output of each step of the exploration. Second, several exploration steps can be spliced on the fly, then users can customize the analysis flow and generate the stratified view on demand according to different tasks and data characteristics.

Local Refinement + Global Filtering. In exploring the network with OnionGraph, each higher-level node is expanded in-situ into lower-level sub-nodes, leading to a local refinement approach. Following the visual information seeking mantra [22], OnionGraph

also implements attribute filters on network nodes and edges to let the user bypass less important information. By a straightforward design, a separate filter can be attached to the profile of each local refinement (abstraction). However, in reality users can hardly remember the detail of each filter. It is hard to restore the network that is filtered entirely, because the filter setting only governs the local network and can not be accessed for any changes. In OnionGraph, we apply a global node and edge filtering mechanism that operates on the entire network. Users specify their filtering rules without selecting a local network in advance. The filtered network is then abstracted according to OnionGraph settings.

3.2 OnionGraph Structure

Figure 2 shows the 5-level hierarchical structure of OnionGraph. The design features the semantic+topological principle: the semantic aggregations (SA) in level-I and level-II are purely semantic, the level-III Relative Regular Equivalence (RRE) combines semantic and topological information, the level-IV Strong Structural Equivalence (SSE) is mostly topological.

In more detail, level-I abstraction groups the original network by node type, level-II abstractions consider the categorical/nominal node attributes. Once a set of attributes are selected, the network is aggregated by grouping all the nodes with the same value on these attributes together. The network links are formed accordingly.

In level-III, RRE method works on the higher-level node groups to extract role sub-groups. Intuitively, RRE considers the neighborhood information of each individual node. The network role is defined recursively in that, the nodes with the same set of roles in their neighborhood are considered with the same network role. In our approach, the initial role partition is constructed through the semantic aggregations, then RRE-defined role can be computed by the set of attribute values in each node's neighborhood.

SSE method in level-IV is similar to RRE. The difference lies in the definition of role partition. SSE requires the network nodes with the same role to have exactly the same set of neighborhoods, more strict than RRE which only considers the role of neighborhoods. In the finest node-level (level-V), each SSE node group is split into individual network nodes, rolling back to the input network granularity.

Built on the 5-level stratified design, OnionGraph provides a wide spectrum of flexibility in customizing the optimized network hierarchies for analysis. The semantic aggregation can apply a manually chosen attribute set which creates sub-hierarchies in level-II. Both RRE and SSE role partitions can specify directed, weighted and fuzzy configurations.

3.3 Hierarchical Focus+Context Exploration

OnionGraph achieves versatile visual analysis capability through the hierarchical focus+context interaction design. As shown in the bottom-right part of Figure 2, by interactive explorations, users can create multiple, hierarchical focuses over the network. Each focused sub-network is associated with an independent abstraction profile. The profile specifies both the current network hierarchy and the abstraction setting. Drill-down and roll-up operations are provided so that the user can manipulate the network hierarchies and customize his own favorite network abstraction for a specific task.

In OnionGraph, each focused sub-network is expanded in-situ in a focus+context manner. The entire network view can juxtapose sub-networks with multiple hierarchy settings. This is quite different from the overview+detail network visualizations and the hierarchy traversing approaches that do not preserve context. More details on these operations are introduced in Section 5.2.

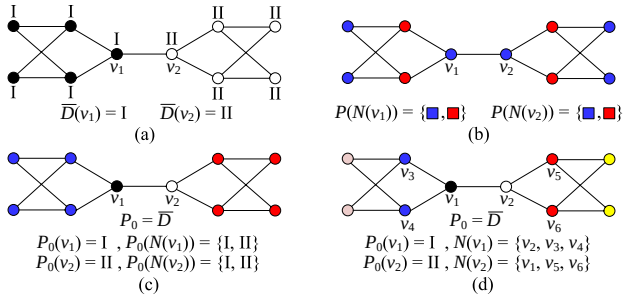


Figure 3: OnionGraph network partitions (undirected case). In each subfigure, node fill color indicates the partition index: (a) semantic aggregation, the selected attribute value is labeled on the node; (b) a regular equivalence partition; (c) the regular equivalence relative to the semantic aggregation in (a); (d) strong structural equivalence.

4 Algorithm

We first formalize the terminologies used throughout the algorithm description.

Heterogenous Network. Let $G = (V,E)$ be a directed and weighted heterogeneous network. $V = \{v_1,...,v_n\}$ and $E = \{e_1,...,e_m\}$ denote the node and link sets. W denotes the adjacency matrix where $w_{ij} > 0$ indicates a link from v_i to v_j, with w_{ij} denoting the link weight. On each node v_i, $N^+(v_i) = \{j|w_{ij} > 0\}$ and $N^-(v_i) = \{j|w_{ji} > 0\}$ indicate the outbound and inbound neighborhood set, both representing its connection pattern. Let $D = \{d_1,...,d_s\}$ be the type and attribute of network nodes in G, with s dimensions in total. $D(v_i) = \{d_1(v_i),...,d_s(v_i)\}$ denotes the type/attribute values of node v_i, with $d_k(v_i)$ indicating the value in the kth dimension.

Network Partition. Let $P : V \to \{1,2,...,t\}$ be a partition (role assignment, coloration or grouping interchangeably) of network G into t sub-groups of nodes. $P(v_i)$ indicates the partition index of node v_i.

The OnionGraph algorithm to create a network abstraction is equivalent to finding a partition of the network according to the abstraction settings. Below we first define various network partitions in OnionGraph, then describe the algorithm implementation to compute such partitions, and finally report its performance.

4.1 Semantic Aggregation

Semantic Aggregation (SA) creates a partition of the network by a selected set of node type or attributes. Formally, for any nodes v_i and v_j in a network G, given the selected attribute set $\overline{D} \subseteq D$, the semantic aggregation network partition P satisfies:

$$\overline{D}(v_i) = \overline{D}(v_j) \Leftrightarrow P(v_i) = P(v_j) \quad (1)$$

Figure 3(a) illustrates a partition based on the node attribute having values "I" or "II". The initial view in OnionGraph is exactly the network aggregated by a primitive node type, e.g., paper/author/venue in the bibliographic network, as shown in Figure 6(a). Multiple network hierarchies can be created when the user adds node attributes to the selected set.

4.2 Relative Regular Equivalence

The original regular equivalence (role equivalence) concept [28] is defined recursively on the network node by the same set of neighborhood roles. For any nodes v_i and v_j in a network G, a regular equivalence network partition P satisfies:

$$P(v_i) = P(v_j) \Rightarrow P(N^+(v_i)) = P(N^+(v_j)) \text{ and } P(N^-(v_i)) = P(N^-(v_j)) \quad (2)$$

However, directly applying regular equivalence on a network will lead to many possible partitions. Figure 3(b) gives a particular case. In the extreme, the identity partition (every node serves

a different role) and the complete partition (every node serves the same role) are both regular. It is hard to find an appropriate regular equivalence partition in real usage. Here we propose a practical solution to apply Relative Regular Equivalence (RRE), which can work on top of the existing semantic aggregation partition. RRE explicitly derives the maximal regular equivalence partition by refining the semantic partition. Mathematically, the RRE partition P over a semantic partition P_0 satisfies:

$$\begin{aligned} &P(v_i) = P(v_j) \Leftrightarrow P_0(v_i) = P_0(v_j) \ \ and \\ &P_0(N^+(v_i)) = P_0(N^+(v_j)) \ \ and \ \ P_0(N^-(v_i)) = P_0(N^-(v_j)) \end{aligned} \tag{3}$$

Figure 3(c) gives an example of the RRE partition relative to the semantic partition in Figure 3(a).

4.3 Strong Structural Equivalence

More stringent to the regular equivalence, Strong Structural Equivalence (SSE) partition [19] requires the network nodes to have exactly the same neighborhood set. For any nodes v_i and v_j in a network G, the SSE network partition P over a RRE partition P_0 satisfies:

$$\begin{aligned} &P(v_i) = P(v_j) \Leftrightarrow P_0(v_i) = P_0(v_j) \ \ and \\ &N^+(v_i) = N^+(v_j) \ \ and \ \ N^-(v_i) = N^-(v_j) \end{aligned} \tag{4}$$

Besides the standard definition, there are a few variations of RRE/SSE partitions. The undirected RRE/SSE (Figure 3) considers the union of inbound and outbound neighborhood sets, the weighted RRE/SSE considers the number of neighborhood role occurrences (RRE) and the weight of the connecting edges (SSE). These options are included in OnionGraph design and configured by user.

4.4 Fuzzy Equivalence

In many practical cases, strict RRE/SSE partitions lead to an extensive number of groups since real-life networks are too complex to have strict structural equivalences. We introduce the fuzzy RRE/SSE partition method which allows a user to control the number of partitions he wants out of a higher-level aggregation.

The first step is to represent each node v_i by its neighborhood vector $R(v_i) = \{c_{i1}, ..., c_{it}, c_{1i}, ..., c_{ti}\}$. For RRE, t is the number of roles out of the upper-level semantic aggregation. For SSE, t is the number of nodes in the network. For unweighted RRE/SSE, $c_{ij}(c_{ji}) = \{0,1\}$ denotes whether the jth role/node is present in the outbound (inbound) neighborhood set of node v_i. For weighted RRE, $c_{ij}(c_{ji})$ denotes the number of occurrence of the jth role in the neighborhood of node v_i. For weighted SSE, $c_{ij}(c_{ji})$ denotes the weight of the edge connecting v_i and v_j. In another setting, the neighborhood vector is normalized by $c_{ij} = c_{ij} / \sum_{j=1,..,t}(c_{ij} + c_{ji})$.

Next, over all the nodes to be partitioned, a pairwise dissimilarity score is computed. Though there are many candidate criteria, we choose the Euclidean distance, because we care the dissimilarity in both orientation and magnitude. Also, the Euclidean distance fits our clustering algorithm well.

Finally, to compute fuzzy equivalence partitions, we apply the k-means clustering algorithm [20]. Note that another possible solution is to set a threshold over the pairwise node similarity score, create a similarity graph by pruning the links below such threshold, and compute the fuzzy groups by graph clustering. However, after a few informal user studies, we found that normal users can hardly understand the threshold setting and the resulting output. In contrast, the only parameter to set in k-means is the number of clusters which is a traceable output of the user interaction.

4.5 Implementation and Performance

For deterministic OnionGraph abstractions, we introduce a unified method to compute the partitions at all five levels. The core concept is the design of a row vector, representing both the semantic and topological information on each node. As shown in Figure 4, the row vector is composed of the node attribute (type) field, the neighborhood relationship, as well as an explicit node identifier when the network is partitioned into individual nodes. The extensions of partition algorithms, e.g., directed and weighted partitions, are supported by design. Finally, the network partition is achieved through an appropriate hash function over the row vectors of all the nodes. The overall implementation has a computational complexity of $O(m+dn)$, where n, m and d are the number of nodes, links and node dimensions of the input heterogeneous network. [3]

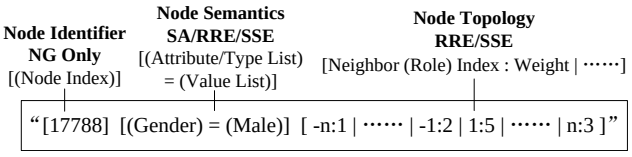


Figure 4: Row vector design in OnionGraph implementation. NG = per-node group.

Table 1: OnionGraph network abstraction performance (SA).

Dataset	Setting	#Node	#Link	Time(s)
VASTC-2012	N/A	2596	36669	0.058
Vis-Bibli.	N/A	20615	106316	0.22
Twitter	N/A	306126	1424427	2.262
Honeypot	N/A	1051595	1158150	12.055

The fuzzy equivalence computation using the k-means clustering has an intrinsic complexity of $O(k \cdot n \cdot d \cdot l)$. k is the number of desired clusters, normally set to a small value for an effective visualization. l is the number of iterations in the computation, generally small for most graphs. n is the number of nodes in the network. d is the maximal number of dimensions of the neighborhood vector. For fuzzy RRE, d equals the number of possible values on the currently selected attributes, and is in most cases bounded. For fuzzy SSE, $d = n$. Therefore, the fuzzy RRE computation is slower than the deterministic version of RRE but still leads to a linear complexity. Fuzzy SSE will be quadratic to the number of nodes and is very slow for large networks. However, during the navigation process, the fuzzy SSE operation can still be viable depending on the size of the sub-network to be expanded.

We evaluate OnionGraph performance in terms of the network abstraction time on a Windows desktop (Quad-core Intel Xeon@3.30GHz with 6GB of memory). Four heterogeneous network data sets from medium to large size are studied: VAST 2012 mini-challenge II traffic network [1](2596 nodes), the bibliographic network of the visualization community (20615 nodes), a Twitter retweet network from KDD Cup 2012 [2](306126 nodes) and a Honeypot security network from the VizSec community [3](1051595 nodes). All the experiments are conducted in the worst-case scenario, i.e., operating over the entire network. 10 trials are issued for each entry and the average time is taken as the result..

The abstraction time of SA, RRE and SSE partitions are given in Table 1, Table 2 and Table 3 respectively. Results suggest that our theoretical analyses correspond well with the actual performance. The completion time of the deterministic version of all three partitions is almost linear to the number of nodes and links, regardless of the directed and weighted setting. The slowest SSE partition completes in 27 seconds on a network with a million nodes and links, still viable for a serious analysis. In contrast, the fuzzy version of the partitions only receive a moderate penalty on RRE, but is too slow for SSE even over a 20000-node network.

[3]In OnionGraph, network data is stored by adjacency lists, so that scanning the links has exactly an $O(m)$ complexity.

Table 2: OnionGraph network abstraction performance (RRE).

Dataset	Setting	#Node	#Link	Time(s)
VASTC-2012	undirected	2596	36669	0.185
Vis-Bibli.	undirected	20615	106316	0.746
Vis-Bibli.	directed	20615	106316	0.561
Vis-Bibli.	weighted	20615	106316	0.46
Vis-Bibli.	fuzzy (#C=5)	20615	106316	1.042
Twitter	undirected	306126	1424427	4.598
Twitter	fuzzy (#C=5)	306126	1424427	7.076
Honeypot	undirected	1051595	1158150	19.902
Honeypot	fuzzy (#C=5)	1051595	1158150	59.083

Table 3: OnionGraph network abstraction performance (SSE).

Dataset	Setting	#Node	#Link	Time(s)
VASTC-2012	undirected	2596	36669	0.253
Vis-Bibli.	undirected	20615	106316	1.157
Vis-Bibli.	directed	20615	106316	1.325
Vis-Bibli.	fuzzy (#C=5)	20615	106316	111.891
Twitter	undirected	306126	1424427	12.203
Honeypot	undirected	1051595	1158150	27.431

5 VISUALIZATION

Figure 1 gives an overview of the OnionGraph interface. It is composed of three parts: OnionGraph network visualization in the center (Section 5.1), the control/filter panel on the left and the legend/list/detail panel on the right (Section 5.2).

5.1 OnionGraph Visual Metaphor

A typical OnionGraph visualization is shown in Figure 5. Each colored node represents a group of original nodes from the underlying network. The node size encodes the number of individual nodes inside the node group. The initial abstraction aggregates all the nodes by their node types ("author", "paper" and "venue" in this figure), as indicated by the icon on the top-right part of each node. The node group in the top-level heterogeneous abstraction is displayed by a filled node, e.g., the spring-green node in the center of Figure 5, representing all 9557 papers. All the other nodes in this figure are expanded from the top-level. They are drawn by the "onion" metaphor composed of several concentric circles. The number of circles indicates the abstraction hierarchy: the semantic aggregation has three circles (venue nodes in Figure 5), RRE has two (author nodes in Figure 5), SSE has one, the individual node only leaves a solid dot. Upon the top-down exploratory analysis, the visual complexity of each node group is reduced as the number of node groups increases, so as to balance the overall complexity of the OnionGraph view.

Node color in OnionGraph is determined by the type/attribute values of each node group. In Figure 5, initially three colors (yellow, spring-green, Indigo) are picked uniformly on color hue. After the expansion of the venue node into sub-nodes, four new colors are assigned with linear hue and saturation offsets from the Indigo color, as shown by the legend in the bottom-left part of Figure 5. Node labels by default display the value of currently selected node type/attributes. When a node group contains only one node, a title is also shown in the label. The selected nodes are drawn with dark-red outlines and labels, coupled with a "+/-" sign upon hovering to indicate the lower/upper level to explore. The neighborhood of the selected nodes and their connecting links are drawn in dark-orange (Figure 1). The link thickness and label encode the number of individual links between the groups. Different from ordinary networks, OnionGraph usually has a loopback link on each node group indicating the internal connection, as shown by the arc above the node.

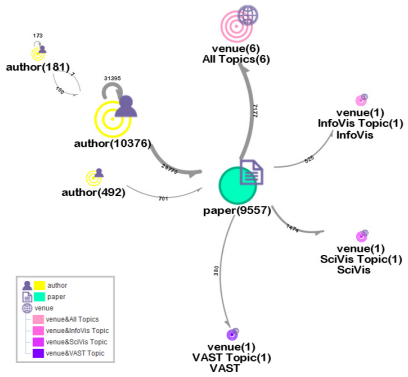


Figure 5: An OnionGraph network visualization of the author-paper-venue bibliographic network in the visualization community. Three author groups indicate different connection patterns: normal authors with co-authors and publications, special authors who only write single-authored papers, and anomalous authors without a publication (potential errors in the data set). Four venue groups indicate the conferences/journals on different topics.

In OnionGraph visual encodings, most critical network measures (e.g. node/link group size) are not linearly mapped to the visual channels. We apply normalizations to favor smaller node/link groups, so that they are still visible given the existence of very large groups. Note that, the normalization is done on a per-type basis for both nodes and links. The largest venue node is of the same size as the largest paper node despite a small number of venues.

5.2 Interaction

OnionGraph interactions allow users to select a portion of interests in the network, specify a different abstraction profile and finally execute to access the finer/coarser-grained visual representation. The network selection is supported in multiple ways, through single-clicks on network nodes, rubber band selection/deselection, and ctrl plus single-click to select a node set with the same abstraction profile. The abstraction profile is configured in the control panel on the left side of the interface. The control panel includes the abstraction level control, attribute multi-selection and several switches indicating the abstraction settings, e.g., directed, weighted and fuzzy RRE/SSE. The selected sub-network is processed by clicking the abstract button. In another usage, users can double-click on the selected node set in the main network view to expand to the next level abstraction. When the context is set to the "collapse" mode, the selected node set is re-grouped into the upper level abstraction.

In the top menu of the OnionGraph interface, configurations such as the layout algorithm, network node/link visual encoding can be accessed. OnionGraph also allows another neighborhood charting mode. As shown in Figure 1, each node group aggregated by RRE is drawn by a chart instead of the onion metaphor, showing the distribution of attribute values in the node's neighborhood. The bottom-left filter panel allows the user to plug in node attribute filters on the network. The OnionGraph abstraction is executed over the filtered network. On the network link, a simplified mechanism is applied. Only filtering over the link type is allowed by a multi-checkbox interface. The rightmost part of the OnionGraph interface shows network details upon visualizations and interactions. The top-right panel displays the node legend indicating the icon/color coding of each node group in the network. The center-right panel displays the list of nodes currently selected in the main view. Upon choosing one node in the list, the node attributes are displayed in a key-value table in the bottom-right panel.

5.3 Network Layout

On OnionGraph layout, we introduce two major algorithms. The first is an improved PivotGraph grid-based layout [27]. The ini-

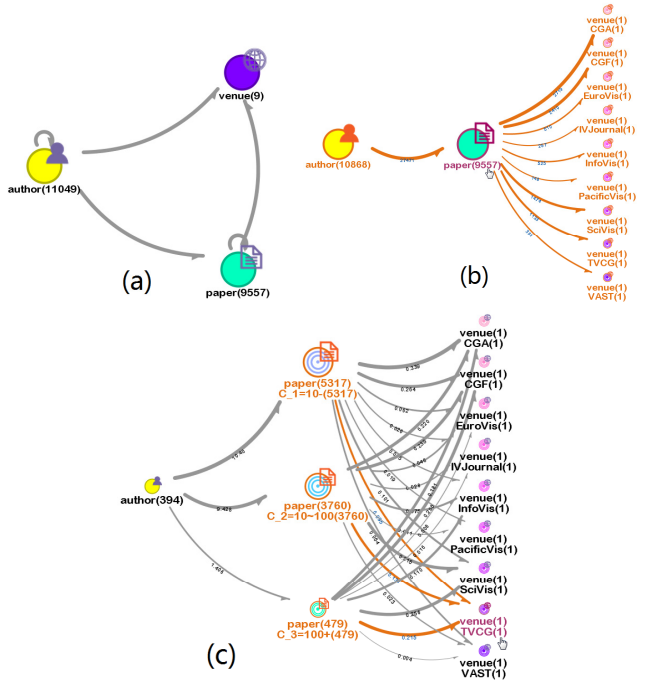


Figure 6: Visualization bibliographic network analysis: (a) Initial view; (b) Venues expanded; (c) Papers expanded by citation categories.

tial PivotGraph layout explicitly selects two node dimensions and places each node by its value on both dimensions. However on OnionGraph, there can be more than one abstraction settings, each managing one part of the heterogeneous network. Applying two global node dimensions of the same abstraction profile can be inadequate for other parts of the network. In our implementation, we improve the algorithm by picking only one global dimension, i.e., the node type, which is mapped to the X axis of the layout. After that, each group of nodes with the same abstraction profile select their own second node dimension on the Y axis. As a result, the OnionGraph abstraction with multiple profiles can have a uniform layout, as shown in Figure 1. When certain part of the network is expanded with a third/more dimension or by RRE/SSE abstraction, the resulting sub-nodes are placed vertically within the grid of the upper-level node. The grid-based layout works for most OnionGraph settings, though does not guarantee an efficient use of space. To compensate this issue, we also implement the classical force-directed layout which optimizes the space utilization and also displays the network topology.

6 Case Study

6.1 Academic Network

We apply OnionGraph to analyze the bibliographic network in the visualization community. The data set was extracted from Arnet-Miner database [25]. It contains the paper information of 9 major visualization conferences and journals, including SciVis, InfoVis, VAST, TVCG [4], etc. Each paper entry includes title, author, publication venue, date, citations, keyword, abstract, etc. We built the heterogeneous bibliographic network with three major node types: 11049 authors, 9557 papers and 9 venues. Five types of links are identified: the co-authorships among authors, the citations pointing from a later paper to an earlier paper, the author-paper affiliation from the author to his papers, the publication from a paper to a venue, and the presentation of an author in a venue (due to the paper), summing up to 106316 links in total. More attributes are extracted on the data set: the papers are classified into 10 topics using Latent Dirichlet Allocation [9] on their title+keyword+abstract contents, with manually assigned topic labels; each author is computed an h-index from his paper citations in the community.

We invited a senior visualization researcher to use our tool to explore the network and gain insights. He was provided with the default overview in the heterogeneous abstraction level showing the relationships among three node types, as shown in Figure 6(a). The link thickness indicates the total number of co-authorships, citations, etc. He proceeded to expand the venue group by venue name under the semantic aggregation level and obtained Figure 6(b). The layout was changed to the grid-based algorithm for clarity. The venues with the highest number of papers are CG&A and CGF, which both publish more than 2000 papers. To analyze the citation performance of these venues, he then expanded the paper node into three sub-nodes by their citation counts: low-citation (< 10), medium-citation ($10 \sim 100$) and high-citation (> 100). He also switched the mapping of link thickness to the average number of links in a link group, e.g., the probability of each paper published in a venue, which is more relevant to the academic performance. From Figure 6(c), he found that though CG&A and CGF published a lot of medium and high-citation papers, their shares in these two groups are lower than those in the low-citation papers. In comparison, SciVis and TVCG (focused node in the figure) have dramatic increases in their shares of medium and high-citation papers. In the same figure, he applied a filter to get rid of the authors with low publications (≤ 10 paper). The resulting author group contains only 394 active people in the community. After that, he conducted analysis on the authors' citation performance. By a lower-level fuzzy RRE abstraction, the active authors were classified into four sub-groups, according to their publication numbers in different citation categories, as shown in Figure 1. The onion metaphors on the author nodes were switched to the neighborhood charts to illustrate their distribution patterns. The sub-group with the largest author icon indicates 309 active authors whose publications include a few low and medium-citation papers and almost only one high-citation paper. The second largest group contains 75 authors with significantly more low/medium-citation papers. The next group (focused node in the figure) probably indicates long-standing fellows in the community, 7 authors who published 41 low, 48 medium and 5 high-citation papers on average. Interestingly, there is another small group (3 authors) who published 75 papers on average, but only 6 of them are medium/high-citation papers.

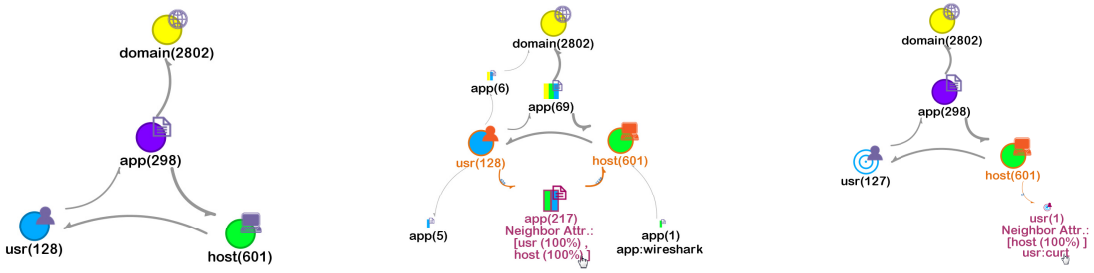
6.2 Host-User-Application Communication

Host-User-Application (HUA) network is generated in a typical lab setting. There are three (four) basic node types: *H* node denotes the host connectivity, which is further partitioned into internal hosts in the Intranet and external domains in the Internet. *U* node denotes the user connectivity chaining (usr). *A* node denotes the application connectivity (app).

We recruited a network administrator to analyze his own lab traces with OnionGraph (we have anonymized the host IP and user ID for privacy issues). He started with the typical HUA network in Figure 7(a). From the graph, he found that there were 128 users logged on 601 internal hosts running 298 unique applications, which connected either internal hosts or 2802 external domains.

He had a few interesting observations when moving from the initial "heterogeneous groups" to "RRE groups" on each node type. First, the app nodes were split into five sub-groups, as shown in Figure 7(b) displayed by neighborhood charts: 1) the majority of apps (217) connected to only internal hosts by users (focused node in the graph); 2) 6 apps connected to only external domains by users; 3) 69 apps connected to both internal hosts and external domains by users; 4) 5 apps did not make network connections; and 5) one app run by an unknown user talked to a few internal hosts. Type-1 apps

[4] TVCG/CGF/CG&A papers belonging to conference proceedings are manually extracted and categorized to their original conference venues.



(a) HUA Overview (b) RRE on app nodes (c) RRE on user nodes

Figure 7: HUA communication network visual analytics.

contain predominantly scientific computing programs while Type-2 and Type-3 have significantly more generic network applications such as ssh, firefox, ftp, etc. In particular, Type-5 node containing only one app (wireshark) is clearly suspicious, possibly leveraged by a malicious user to sniff packets on the network. Second, the user node had been divided into two groups (Figure 7(c)): 127 users that had run apps to connect to other computers; and the only user who never ran apps. The Type-1 users are primarily enterprise users who are allowed to run scientific programs. The Type-2 user is the system administrator. It is clear that normal users and privileged users have distinguished activity patterns.

7 Conclusion

OnionGraph is a visual analysis framework for the exploration of large heterogeneous networks. It is realized by scalable algorithms creating attribute-based aggregations and various structural equivalence network partitions. By combining semantic and topological information for a hierarchical abstraction, OnionGraph enables the level-of-detail viewing of heterogeneous networks. The navigation and filtering interactions in complement to each other are shown to be effective in flexibly controlling the analysis process with OnionGraph. Evaluation results in case studies demonstrate that OnionGraph is useful in many heterogenous network analysis scenarios where the task is exploratory and involves both entity-centric and structural problem-solving.

References

[1] IEEE VAST Challenge, 2012. `http://www.vacommunity.org/VAST+Challenge+2012`.

[2] KDD Cup, 2012. `http://www.kddcup2012.org/`.

[3] VizSec community, 2013. `http://vizsec.org`.

[4] J. Abello, F. van Ham, and N. Krishnan. ASK-GraphView: A large scale graph visualization system. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):669–676, 2006.

[5] D. Archambault, T. Munzner, and D. Auber. Multi-level graph layout by topological features. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):305–317, 2007.

[6] D. Archambault, T. Munzner, and D. Auber. Grouseflocks: Steerable exploration of graph hierarchy space. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):900–913, 2008.

[7] D. Auber, Y. Chiricota, F. Jourdan, and G. Melancon. Multiscale visualization of small world networks. In *IEEE Symposium on Information Visualization (InfoVis'03)*, pages 75–81, 2003.

[8] A. Bezerianos1, F. Chevalier, P. Dragicevic, N. Elmqvist, and J. D. Fekete. Graphdice: A system for exploring multivariate social networks. *Computer Graphics Forum*, 29(3):863–872, 2010.

[9] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[10] N. Cao, J. Sun, Y.-R. Lin, D. Gotz, S. Liu, and H. Qu. Facetatlas: Multifaceted visualization for rich text corpora. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1172–1181, 2010.

[11] C. Dunne, N. H. Riche, B. Lee, R. Metoyer, and G. Robertson. Graphtrail: analyzing large multivariate, heterogeneous networks while supporting exploration history. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1663–1672, 2012.

[12] N. Elmqvist and J.-D. Fekete. Hierarchical aggregation for information visualization: Overview, techniques and design guidelines. *IEEE Transactions on Visualization and Computer Graphics*, 16(3):439–454, 2010.

[13] E. Gansner, Y. Koren, and S. North. Topological fisheye views for visualizing large graphs. In *IEEE Symposium on Information Visualization (InfoVis'04)*, 2004.

[14] J. Heer and D. Boyd. Vizster: visualizing online social networks. In *IEEE Symposium on Information Visualization (InfoVis'05)*, pages 32–39, 2005.

[15] H. Kang, C. Plaisant, B. Lee, and B. B. Bederson. Netlens: Iterative exploration of content-actor network data. *Information Visualization*, 6(1):18–31, 2007.

[16] J. Lamping, R. Rao, and P. Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *Proceedings of the international conference on Human factors in computing systems (CHI'95)*, 1995.

[17] B. Lee, C. Plaisant, C. S. Parr, J.-D. Fekete, and N. Henry. Task taxonomy for graph visualization. In *Proceedings of BEyond time and errors: novel evaLuation methods for Information Visualization (BELIV)*, pages 82–86, 2006.

[18] B. Lee, G. Smith, G. Robertson, M. Czerwinski, and D. S. Tan. Facetlens: exposing trends and relationships to support sensemaking within faceted datasets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1293–1302, 2009.

[19] F. Lorrain and H. C. White. Structural equivalence of individuals in social networks. *The Journal of Mathematical Sociology*, 1(1):49–80, 1971.

[20] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press*, 1:281–297, 1967.

[21] 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.

[22] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the IEEE Symposium on Visual Languages*, pages 336–343, 1996.

[23] B. Shneiderman and A. Aris. Network visualization by semantic substrates. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):733–740, 2006.

[24] G. Sindre, B. Gulla, and H. G. Jokstad. Onion graphs: aesthetics and layout. In *Proceedings of the IEEE Workshop on Visual Languages*, pages 287–291, 1993.

[25] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su. Arnetminer: Extraction and mining of academic social networks. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 990–998, 2008.

[26] F. van Ham and A. Perer. "search, show context, expand on demand": Supporting large graph exploration with degree-of-interest. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):953–960, 2009.

[27] M. Wattenberg. Visual exploration of multivariate graphs. In *SIGCHI conference on Human Factors in computing systems (CHI'06)*, 2006.

[28] D. R. White and K. P. Reitz. Graph and semigroup homomorphisms on networks of relations. *Social Networks*, 5(2):193–234, 1983.