# A Visual-Analytic Toolkit for Dynamic Interaction Graphs

X. Yang, S. Asur, S. Parthasarathy
Department of Computer Science
Ohio State University
{yangxin,asur,srini}@cse.ohio-state.edu

S. Mehta
IBM India Research Lab
New Delhi, India 110070
sameepmehta@in.ibm.com

In this article we describe a visual-analytic tool for the interrogation of evolving interaction network data such as those found in social, bibliometric, WWW and biological applications. The tool we have developed incorporates common visualization paradigms such as zooming, coarsening and filtering while naturally integrating information extracted by a previously described event-driven framework for characterizing the evolution of such networks. The visual front-end provides features that are specifically useful in the analysis of interaction networks, capturing the dynamic nature of both individual entities as well as interactions among them. The tool provides the user with the option of selecting multiple views, designed to capture different aspects of the evolving graph from the perspective of a node, a community or a subset of nodes of interest. Standard visual templates and cues are used to highlight critical changes that have occurred during the evolution of the network. A key challenge we address in this work is that of scalability – handling large graphs both in terms of the efficiency of the back-end, and in terms of the efficiency of the visual layout and rendering. Two case studies based on bibliometric and Wikipedia data are presented to demonstrate the utility of the toolkit for visual knowledge discovery.

## 1. INTRODUCTION

Important scientific discoveries almost always rely on visual confirmation – from Galileo seeing the moons of Jupiter to Ginnig and Rohrer seeing atoms on a surface. Visual aids and interactivity are often key to forming important insights, particularly when targeting hard problems. Given the nature of the knowledge discovery process with a human-in-the-loop a visual analytic interactive front-end is often key to effective information synthesis.

In this article we present such a visual analytic toolkit targeted toward the analysis of dynamic interaction networks.Many real world problems can be modeled as complex interaction networks where nodes represent entities of interest and edges mimic the interactions or relationships among them. Fueled by technological advances and inspired by empirical analysis, the number of such problems and the diversity of domains from which they arise – physics, sociology, technology, biology, chemistry, metabolism and nutrition – is growing steadily. In a large number of such domains the networks governing interactions are known to evolve or change – bibliometric data, social network data, epidemiology data, biological networks, and the World Wide Web to name a few examples.



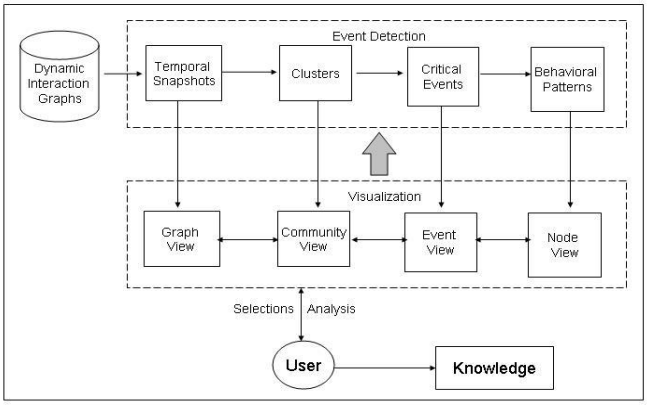


**Figure 1: Overview of proposed toolkit**

In such networks, the addition and deletion of edges and nodes can be used to represent changes in the interactions among the modeled entities. The challenge is to identify and localize the portions of the network that are changing to help characterize the type of change and its potential causes, *visually*. A related challenge is to facilitate interactive interrogation, i.e., the user needs to be able to interactively select and zoom down to clusters, entities of interest, as well as specific dynamic interactions and events that govern the evolution of interaction networks over time.

To address these challenges, we introduce a visual toolkit specifically designed to analyze dynamic graphs. Figure 1 provides a schematic representation of the components of our proposed visual analysis toolkit. The back-end of our toolkit leverages a previously developed event-detection framework for analyzing dynamic interaction networks [2]. This framework presents a methodology to detect critical events affecting nodes and communities within such networks and offers a principled way to characterize their evolution through the composition of various incrementally computable behavioral indices such as stability, sociability and influence. As shown in the figure, this information is tightly integrated with our visual front end to provide a highly interactive interface for

the end-user.

To facilitate visual analysis, the front end of the toolkit presents the user with the option of multiple views - a *graph view* which is a cumulative snapshot representation of the graph at different points in time, a *community view* which represents the cluster arrangements of the snapshot graphs, an *event view* which demonstrates the transformations that have occurred over time, and a *node view* which details the evolutionary behavior of individual entities. We allow the user to pick the intervals of interest and drill down onto the corresponding events and behavioral measures within that time-frame. We use a weighting function to associate different behavioral characteristics such as influence and sociability with nodes and importance and recency with edges. These weights are then mapped onto effective visual cues to localize features of interest. Overall, the front-end conforms to the popular mantra – *overview, zoom, filter and details on demand* [5].

For exploratory visual analysis, timeliness of the computation and presentation is important. This is of particular importance when one is considering large real-world graphs such as social networks like Myspace and Flickr. Even simple layout and plotting tools suffer when the size of the graph is very large. For our toolkit, we make use of key optimizations to speed up computation in the back-end, and leverage the use of coarsening mechanisms to provide scalable performance in the front-end to squeeze relevant information in the available pixel space.

In short, the challenges that we address in our work are:

1. *Identifying interesting properties of interactions among nodes and communities such as stability, popularity, frequency etc*

2. *Analysis of communities over time to discovers changes that occur with respect to other nodes and communities*

3. *Analysis of relationships of a node with its neighbors to discover trends in its importance*

4. *Ensure scalability to large graphs and facilitate interactive exploratory visual analysis*

We present two case studies on real evolving graph datasets to underline some of the benefits of our toolkit for visual analysis.

## 2. RELATED WORK

Recently, there has been considerable interest in analyzing dynamic interaction graphs. Leskovec *et al* [14] studied the evolution of graphs based on various topological properties, such as the degree distribution and small-world properties of large networks and proposed the Forest-Fire graph generation model. Backstrom *et al* [3] studied formation of groups and the ways they grow and evolve over time. Chakrabarti *et al* [6] introduced evolutionary clustering which involves incrementally obtaining high-quality clusters for a set of objects. They proposed evolutionary settings for the kmeans and agglomerative hierarchical clustering algorithms. Falkowski *et al* [7] have studied the evolution of communities that are stable or fluctuating based on subgroups. Sun and others [21] have proposed GraphScope for parameter-free pattern mining of time-evolving graphs. In the context of event-based feature analysis Samtaney *et al* [19] described an approach for extracting coherent regions from 2-dimensional and 3-dimensional scalar and vector fields for tracking purposes. To study the evolution of these regions over time, they present certain evolutionary events for objects. Event-based methods have also been applied on spatial data [22] and clustered stream data [20].

There has been considerable amount of work in visualization of social networks. Heer and Boyd [10] have developed the *Vizster* tool for visualizing online social networks. The toolkit can be used to explore communities, linkage and supports keyword search. Perer and Schneiderman [16] have presented a general social network visualization toolkit. The toolkit supports ranking of nodes based on various properties of the graph like centrality, cut-points etc. Abello and others [1] have presented a graph visualization toolkit called ASK–GraphView. The toolkit uses a clustering algorithm to construct a hierarchy which is easy to browse. Henry and Fekete [11] have presented a dual representation for visualizing social networks. The proposed toolkit *MatrixExplorer* uses a synchronized graph and matrix representation of the network for visualization. A key difference separating our work from the above methods is that they are designed to operate primarily on static interaction graphs.

Gloor and Zhao [9, 8] have developed *iQuest*, a visual toolkit to understand topics of discussion among actors in a semantic web. Kumar and Garland [13] have presented algorithms to visualize a graph in hierarchical fashion by exploiting existing correlations. Time-varying graphs are handled by producing animations composed of static snapshots. Qeli *et al.* [17] have proposed algorithms to visualize time-varying matrices. The matrices used in the article represent clustering results. The authors generate a cumulative matrix and use colors to denote changes in memberships. The toolkit can also be used to find a group of elements which are part of the same cluster for an extended period of time. The community view presented in our work serves a similar purpose. We also provide views showing changes to nodes, neighborhoods and communities across time. The tight integration of our event detection back-end is a key difference from the above methods.

## 3. BACKGROUND: EVENT DETECTION

This section provides the reader with a synopsis of our previous work in order that this document be self contained. Additional details can be found elsewhere [2]. We define a temporal snapshot $S_i = (V_i, E_i)$ of a graph $G = (V, E)$ to be a graph representing only entities and interactions active in a particular time interval $[T_{s_i}, T_{e_i}]$, called the snapshot interval.

As the graph evolves, its dynamic behavior over time can be represented as a set of $S$ non-overlapping temporal snapshots. We use clusters of the graph to represent its structure at different snapshots. We believe that studying the evolution of these clusters, in particular their formation, transitions and dissolution, can be extremely useful for effectively characterizing the corresponding changes to the network over time. Accordingly, each $S_i$ is partitioned into $k_i$ communities or clusters denoted by $C_i = \{C_i^1, C_i^2, \ldots, C_i^{k_i}\}$. Let $S_i$ and $S_{i+1}$ be snapshots of $S$ at two consecutive time intervals with $C_i$ and $C_{i+1}$ denoting the set of clusters respectively. The critical events we define on communities or clusters are:

**1) Continue:** A cluster $C_{i+1}^j$ is marked as a continuation of $C_i^k$ if $V_{i+1}^j$ is the **same** as $V_i^k$ (i.e their vertex sets are the same). Note that we do not impose the constraint that the edge sets should be the same.

$$Continue(C_i^k, C_{i+1}^j) = 1 \text{ iff } V_i^k = V_{i+1}^j$$

**2) $\kappa$-Merge:** Two different clusters $C_i^k$ and $C_i^l$ are marked as merged if there exists a cluster in the next timestamp that contains at least $\kappa\%$ of the nodes belonging to these two clusters. The essential condition for a merge is :
$Merge(C_i^k, C_i^l, \kappa) = 1$ iff $\exists C_{i+1}^j$ such that

$$\frac{|(V_i^k \cup V_i^l) \cap V_{i+1}^j|}{Max(|V_i^k \cup V_i^l|, |V_{i+1}^j|)} > \kappa\%$$

and $|V_i^k \cap V_{i+1}^j| > \frac{|C_i^k|}{2}$ and $|V_i^l \cap V_{i+1}^j| > \frac{|C_i^l|}{2}$. This condition will only hold if there exist edges between $V_i^k$ and $V_i^l$ in timestamp $i+1$. Intuitively, it implies that new interactions have been created between nodes which previously were part of different clusters. We allow the user the option of varying the $\kappa$ parameter in the visual interface.

**3) $\kappa$-Split:** A single cluster $C_i^j$ is marked as split if $\kappa\%$ of nodes from this cluster are present in 2 different clusters in the next timestamp. The essential condition is that:
$Split(C_i^j, \kappa) = 1$ iff $\exists C_{i+1}^k, C_{i+1}^l$ such that

$$\frac{|(V_{i+1}^k \cup V_{i+1}^l) \cap V_i^j|}{Max(|V_{i+1}^k \cup V_{i+1}^l|, |V_i^j|)} > \kappa\%$$

and $|V_{i+1}^k \cap V_i^j| > \frac{|C_{i+1}^k|}{2}$ , $|V_{i+1}^l \cap V_i^j| > \frac{|C_{i+1}^l|}{2}$.

Intuitively, a split signifies that the interactions between certain nodes are broken and not carried over to the current timestamp, causing the nodes to part ways and join different clusters.

**4) Form:** A new cluster $C_{i+1}^k$ is said to have been formed if none of the nodes in the cluster were grouped together at the previous time interval i.e. no 2 nodes in $V_{i+1}^k$ existed in the same cluster at time period $i$.

$Form(C_{i+1}^k) = 1$ iff $\exists$ no $C_i^j$ such that $V_{i+1}^k \cap V_i^j > 1$
Intuitively, a form indicates the creation of a new community or new collaboration.

**5) Dissolve:** A single cluster $C_i^k$ is said to have dissolved if none of the vertices in the cluster are in the same cluster in the next timestamp i.e. no two entities in the original cluster have an interaction between them in the current time interval.

$Dissolve(C_i^k) = 1$ iff $\exists$ no $C_{i+1}^j$ such that $V_i^k \cap V_{i+1}^j > 1$

Intuitively, a dissolve indicates the lack of contact or interactions between a group of nodes in a particular time period. Events associated with individual nodes are described next.

**6)Join:** A node is said to join cluster $C_i^j$ if it exists in the cluster at timestamp $i$ and it was not present in a similar cluster in the previous timestamp.
$Join(v, C_i^j) = 1$ iff $\exists C_i^j$ and $C_{i-1}^k$ such that $C_{i-1}^k \cap C_i^j > \frac{|C_{i-1}^k|}{2}$ and $v \notin V_{i-1}^k$ and $v \in V_i^j$

The cluster similarity condition ensures that $C_i^j$ is not a newly formed cluster.

**7)Leave:** A node is said to leave cluster $C_{i-1}^k$ if it no longer is present in a cluster with most of the nodes in $V_{i-1}^k$.
$Leave(v, C_i^j) = 1$ iff $\exists C_i^j$ and $C_{i-1}^k$ such that $C_{i-1}^k \cap C_i^j > \frac{|C_{i-1}^k|}{2}$ and $v \in V_{i-1}^k$ and $v \notin V_i^j$
The similarity constraint between the two clusters is used to maintain cluster correspondence.

### *Behavioral Analysis:.*

We use the Join and Leave events, described above, to define four behavioral measures that can be *incrementally computed* at each time interval using the events discovered in the current interval.

**Stability Index:** The Stability index measures the tendency of a node to have interactions with the same nodes over a period of time. A node is highly stable if it belongs to a very stable cluster (one with infrequent joins and leaves). Let $cl_i(x)$ represent the cluster that node $x$ belongs to in the $i^{th}$ time interval. The Stability Index (SI) for node $x$ over $T$ timestamps is measured incrementally as:

$$SI(x,T) = \sum_{i=1}^{T} \frac{|cl_i(x)|}{\sum_{j=1}^{V_i}(Leave(j, cl_i(x)) + Join(j, cl_i(x)))}$$

**Sociability Index:** A related measure is the Sociability Index, which is a measure of the number of *different* interactions that a node participates in. Let $cl_i(x)$ be the cluster that node $x$ belongs to at time $i$. Then, the Sociability Index is defined as:

$$SoI(x) = \frac{\sum_{i=1}^{T-1}(Join(x, cl_{i+1}(x)) + Leave(x, cl_i(x)))}{|\text{Activity(x)}|}$$

and $|\text{Activity(x)}| > Min\_activity$

where $Activity(x) = \sum_{i=1}^{T}(x \in V_i)$ indicates the number of intervals that node $x$ is active. The measure gives high scores to nodes that are involved in interactions with **different** groups. The threshold $Min\_activity$ corresponds to the minimum number of active intervals for a node to be considered sociable. [1]

**Influence Index:** The influence index of a node is a measure of the influence this node has on others. We would like to find nodes that influence other nodes into participating in critical events. The intuition is that, if a large number of nodes leave or join a cluster with high frequency when a certain node $x$ does, it suggests that node $x$ has a certain positive influence on the movement of the others. Let $Companions(x)$ represent all nodes over all timestamps that join or leave clusters with node $x$. The Influence for node $x$ is given by:

$$Inf(x) = \frac{|\text{Companions(x)}|}{|\text{Moves(x)}|}$$

Here $Moves(x)$ represents the number of *Join* and *Leave* events $x$ participates in. Note that, this definition by itself, does not measure influence, since nodes that interact and move along with highly influential nodes will have high Influence score values as well. Such nodes are down-weighted accordingly as described previously[2].

[1]We used a $Min\_activity$ value of 1/2 the number of time intervals, for our experiments.

**Popularity Index:** The Popularity Index of a cluster at time interval $[i, i+1]$ is a measure of the number of nodes that are attracted to it during that interval. It is defined as:

$$PI(C_i^j) = (\sum_{x=1}^{V_i} Join(x, C_i^j)) - (\sum_{x=1}^{V_i} Leave(x, C_i^j)) \quad (1)$$

This measure can be used to highlight clusters that undergo significant transformation over the course of a time interval.

# 4. DATASETS

## 4.1 DBLP Dataset

We used a subset of the DBLP bibliography [2] to generate a co-authorship network representing authors publishing in several important conferences in the field of AI, databases and data mining. We chose all papers over a 10 year period (1997-2006) that appeared in 28 key conferences spanning mainly these three areas. We converted this data into a co-authorship graph, where each author is represented as a node and an edge between two authors corresponds to a joint publication by these two authors. We chose the snapshot interval to be a year, resulting in 10 consecutive snapshot graphs, containing 23136 nodes and 54989 edges. It has been shown that collaboration networks display many of the structural features of social networks[12, 15]. Hence, this is a good representative dataset for this study.

## 4.2 Wikipedia Dataset

The Wikipedia online encyclopedia is a large collection of webpages providing comprehensive information concerning various topics. The dataset we employ represents the Wikipedia revision history and was obtained from Berberich [4]. It consists of a set of webpages as well as links among them. It comprises of the editing history from January 2001 to December 2005. The temporal information for the creation and deletion of nodes (pages) and edges (links) are also provided. We chose a large subset of the provided dataset, consisting of 779005 nodes (webpages) and 32.5 M edges. We constructed snapshots of 3 month intervals, and considered the first 6 snapshots for our analysis.

# 5. OPTIMIZATIONS FOR FAST EVENT DETECTION

The event detection proceeds in an iterative manner, with every two successive snapshots analyzed to compute events among them. So, at each stage, we analyze the respective clusters of $T_i$ and $T_{i+1}$ and compute events between them. First, it is important to note that, since we will be considering only a pair of timestamps at a time, we do not need to consider all $N$ nodes, since many of the nodes may not be active over the time period. Hence, for event detection between $T_i$ and $T_{i+1}$, we need to examine only the nodes active over either of the two timestamps. This greatly reduces the complexity of the event detection algorithm. Table 2 gives the percentage of active nodes, for both datasets. It can be observed that the percentage of active nodes for a pair of snapshots never increases beyond 40% of the total number of nodes.

[2]http://www.informatik.uni-trier.de/ ley/db/

| Time Stamp | # of clusters (DBLP) | # of clusters (Wikipedia) |
|---|---|---|
| 1 | 869 | 297 |
| 2 | 950 | 1620 |
| 3 | 955 | 4783 |
| 4 | 865 | 9830 |
| 5 | 1057 | 12085 |
| 6 | 805 | 18318 |
| 7 | 1112 | |
| 8 | 1166 | |
| 9 | 1434 | |
| 10 | 1080 | |

**Table 1: Number of clusters.**

To facilitate exploratory visual analysis, we need to ensure that event detection can be performed quickly, as the events need to be shown to the user for further analysis. Our detection algorithm relies on finding intersections and unions of the cluster sets, as evident from the formulae presented in the previous section. When the number of clusters is large, finding these intersections and unions can be expensive even with the bit matrix operations we described in our earlier work[2]. Finding the intersection between $k_i$ clusters of $T_i$ and $k_{i+1}$ clusters of $T_{i+1}$ has time complexity $O(k_i * k_{i+1})$; For most real-world graphs, the number of communities can be quite large ($k_i$*$k_{i+1}$>$N$). The number of clusters obtained for each timestamp of the DBLP and Wikipedia graphs are shown in Table 1.

To enhance the performance of the back-end, particularly when scaling to datasets like the Wikipedia data, we develop an optimization to calculate the cluster intersection matrix $I$ in $O(M)$ time, where $M$ is the number of nodes active in either $T_i$ or $T_{i+1}$ ($M$<=$N$). The idea is as follows. We first construct two cluster vectors (for the two timestamps considered), to represent the clusters (community) that a node belongs to in a timestamp. We then traverse these vectors sequentially and update the cluster intersection matrix $I$, as shown in Algorithm 1.

**Algorithm 1** Intersection$(C_i, C_{i+1})$

**Input:** Set of $M$ active nodes
**for** $m = 1$ to $M$ **do**
  $cluster_i[m]$=cluster id that node $m$ belongs to in timestamp $T_i$
  $cluster_{i+1}[m]$=cluster id that node $m$ belongs to in timestamp $T_{i+1}$
**end for**
// We then traverse these cluster vectors from left to right.
**for** $m = 1$ to $M$ **do**
  **if** $m$ is active in $T_i$ and $T_{i+1}$ **then**
    $I[cluster_i[m]][cluster_{i+1}[m]] + +$;
  **end if**
**end for**

The cluster unions can be computed easily by taking the sum of the cluster sizes and subtracting the intersection obtained from $I$.

Note that all the behavioral measures described above can be computed incrementally. We maintain a vector in memory for each of the behavioral indices. As the increments are computed for each timestamp, the corresponding values are updated. Thus, at any given time point, one can obtain the Index values in a straightforward manner. These measures are displayed to the user as part of the node view, which will be described in the next section.

| | DBLP | | Wikipedia | |
|---|---|---|---|---|
| Time stamps | Active Nodes | Time (secs) | Active Nodes | Time (secs) |
| 1-2 | 0.23 | 0.088 | 0.03 | 0.12 |
| 2-3 | 0.25 | 0.094 | 0.07 | 0.5 |
| 3-4 | 0.24 | 0.087 | 0.13 | 1.7 |
| 4-5 | 0.26 | 0.099 | 0.19 | 4.5 |
| 5-6 | 0.27 | 0.091 | 0.22 | 11.15 |
| 6-7 | 0.29 | 0.096 | | |
| 7-8 | 0.34 | 0.12 | | |
| 8-9 | 0.41 | 0.14 | | |
| 9-10 | 0.40 | 0.14 | | |

**Table 2: Computation Times for the Back End.**

| Time stamps | Old[2] (secs) | Optimized (secs) |
|---|---|---|
| 1-2 | 10.25 | 0.12 |
| 2-3 | 90.92 | 0.5 |
| 3-4 | 704.93 | 1.7 |
| 4-5 | 2256.34 | 4.5 |
| 5-6 | 5016.52 | 11.15 |

**Table 3: Computation Time Comparison.**

The timing results for the event detection and index computation are given in Table 2. To emphasize the savings, we also present the performance of our earlier implementation[2] on the Wikipedia dataset, without the above mentioned optimizations (see Table 3). In a nutshell the optimizations are very effective and ensure that the back-end is significantly faster than before and is within very reasonable limits given the scale of the data being operated on.

# 6. VISUAL ANALYSIS

In this section, we highlight the key components of the interface along with associated user interaction features. We also motivate the need for the components towards the overall goal of knowledge extraction from evolving graphs. The key components of the toolkit are:

**Data Loader:** This component is used for reading the input data and label files. The data to be read is in the form of temporal snapshot graphs as we described earlier in this section. Each graph corresponds to one time step and is stored in an **edge file** format. Additionally, a label file is read which associates each node in the graph with a unique identifier and name, if available. If clusters are available, then we provide an option to read the cluster file as well. If not, clustering can be performed online. We provide options for kMetis or MCL clustering. Once the data is read, preprocessing is done to create the cluster vectors for the first two timestamps.

**View Mode Selector:** Once the data is ready, the user selects one of the four supported views and the relevant parts of the interface get activated. Before detailing the views, we describe our hierarchical representation.

**Coarsening:** To visualize large graphs on the screen, we choose to coarsen the graphs, using the cluster information to construct multi-level hierarchies of nodes. This facilitates easy visual interpretation, since it provides the user the ability to consider and drill down to sections of interest in the graph. At the lowest level in the hierarchy, we have the nodes and edges of the graph. The graph is clustered to produce *base clusters*. These clusters are then further clustered internally by our coarsening algorithm into multiple levels. The kMetis algorithm is employed in this stage. Before coarsening, a new edge file is created by transferring edges between nodes of different clusters to the corresponding *supernodes* representing these clusters. This edge file is used by kMetis to obtain the higher level supernodes. Thus each level consists of a graph of supernodes, each of which represents a cluster of lower-level nodes. The user is initially provided with a high level view in the form of connected supernodes. Also, the physical sizes of the supernodes in the interface reflect the sizes of the cluster they represent. Clusters that contain a large number of nodes can thus be differentiated from singleton clusters with ease. Dynamic behavioral information about the nodes and clusters are also provided, as we will describe below. The user can then select one or a group of interesting supernodes to drill down and visualize the corresponding section of the original graph.

**Graph View**– In this view, the entire dynamic network is displayed as a graph [3]. As mentioned above, the graph is presented as a multi-level hierarchy. The bottom-most level represents the graph itself in the form of nodes and edges. The level immediately above represents supernodes, where each node is a cluster of the lowest-level nodes. Each supernode in this level is labeled with the *Popularity Index* value of the cluster it corresponds to. It is also colour-coded to reflect the strength of the *Popularity Index* values. The user can select an interesting set of clusters, and descend to the lower level to visualize the nodes in question. In our implementation, the sequence of colors for nodes (from low weight to high weight) is *Dark Yellow, Light Yellow, Light Green, Dark Green*. Similarly, for edges the sequence is *Dark Red, Light Red, Light Blue, Dark Blue*. The progression of colors for nodes and edges is shown in the bottom right corner of the visual interface. At the lowest level, properties of nodes - *sociability, stability and influence* are computed as described in Section 3 to assign a weight. Finally, the weights are normalized between [0,1] and are mapped to a color which is then used to render the graphs. We also provide a facility for multi-weighting a node, where we compute the weight taking into account two of these behavioral measures. This is beneficial for discovering correlations among properties of nodes. The relative importance of each edge is primarily captured by its *temporal stability*, i.e., for how many **consecutive** time steps that particular edge is observed. Note that, the importance of an edge (interaction) in terms of these measures can be determined based on the nodes involved. For instance, the stability of the edge can be represented as the product of the stability indices of the two nodes.

$$SI(x, y, T) = SI(x, T) * SI(y, T)$$

Moreover, to give less importance to old edges (which are not observed recently), we use different line styles. For example, if an edge also occurred in the previous time stamp, we use a dashed line to represent temporal stability. Edges that were not observed recently are represented by a straight line.

**Community View**– This view displays various clusters or communities present in the network. Once the user selects this view, the system presents the user with the clusters that the nodes belong to. The membership of nodes to the

[3]We use the Graphviz(http://www.graphviz.org/) layout tool to visualize the graph.

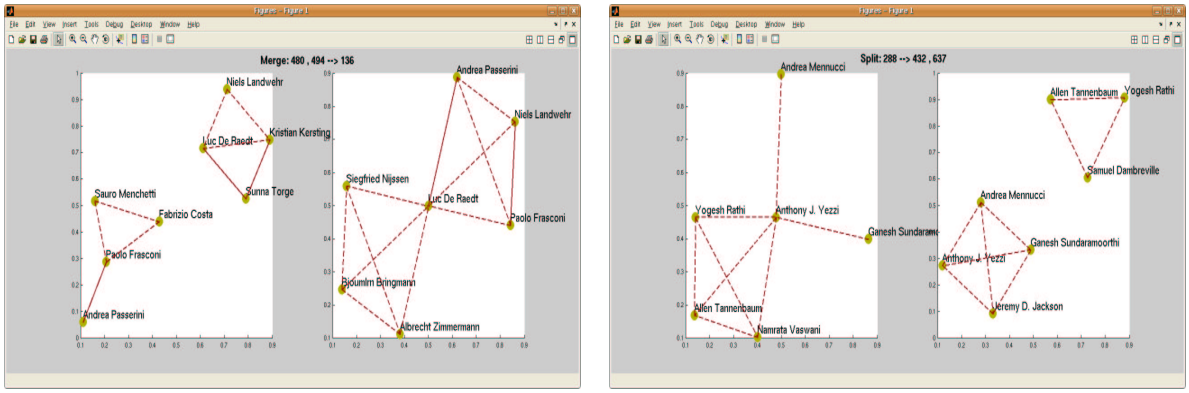


**Figure 3: Examples of a) Merge Event b) Split Event**

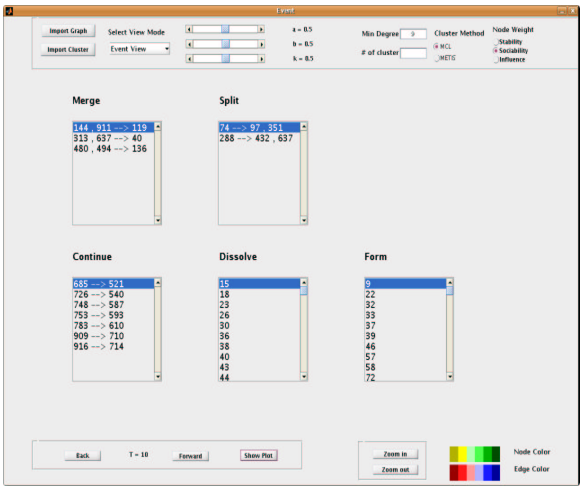


**Figure 2: Event View for DBLP**

clusters are taken into account by using the same color and same marker for rendering.

**Event View**– The event view is designed to provide information regarding transformations that occur in the graph over time. This view displays a set of all critical events that occur between the current and previous intervals. The user can choose different time intervals and observe the events that transpire among them. Figure 2 shows the set of events between clusters of time stamp 9 and 10. At the top of the GUI, there are three bars $a$, $b$ and $k$ which correspond to the $\alpha$,$\beta$ and $\kappa$ parameters for the event detection algorithm described earlier. The user can vary these parameters and examine the critical events obtained. In the middle of the screen, the GUI provides a list of all critical events observed. The user can select one of these events, which provides details on the nodes and clusters involved. An example each of a Merge and a Split event are shown in Figure 3. As it can be observed, the detailed representation of the event is also visualized on the screen giving the user a representation of the nodes involved and the change that has occurred. For the Merge and Split events in the Event View, we also provide a *Semantic Similarity* ranking. This is of use for graphs that have associated category or term hierarchy information. To begin with, the Information Content (IC) of a term (category or keyword-set), using Resnik's definition [18], is given as:

$$IC(k_i) = -ln\frac{F(k_i)}{F(root)} \tag{2}$$

where $k_i$ represents a term and $F(k_i)$ is the frequency of encountering that particular term over all the entire corpus. Here, $F(root)$ is the frequency of the root term of the hierarchy. Note that frequency count of a term includes the frequency counts of all subsumed terms in an is-a hierarchy. Also note that terms with smaller frequency counts will therefore have higher information content values (i.e. more informative). Using the above definition, the Semantic Similarity (SS) between two terms (categories) can be computed as follows:

$$SS(k_i, k_j) = IC(lcs(k_i, k_j)) \tag{3}$$

where $lcs(k_i, k_j)$ refers to the lowest common subsumer of terms $k_i$ and $k_j$. To define the semantic similarity between two clusters, one can employ an information theoretic mutual information measure. Given probabilities of terms $m$ and $n$ occurring in a cluster as $p(m)$ and $p(n)$ respectively, and their co-occurrence probability $p(mn)$, the Semantic Mutual Information (SMI) between the two clusters $C_i^a$ and $C_j^b$ can be given as:

$$SMI(C_i^a, C_i^b) = \sum_{m=1}^{k^a}\sum_{n=1}^{k^b} SS(m,n) * p(mn) * \log_{k^a * k^b}\frac{p(mn)}{p(m) * p(n)} \tag{4}$$

However, while this measure accurately captures similarities, it is not very scalable for graphs with large category hierarchies, due to the amount of computation required and memory consumed. In these cases, the semantic similarity between two clusters can be computed as:

$$Inter\_SS(C_i^a, C_i^b) = \frac{\sum_{k^a}^{m=1}\sum_{k^b}^{n=1} SS(m,n)}{k^a * k^b} \tag{5}$$

Note that, the semantic similarity values between terms are pre-computed, while computing the Inter_SS() of clusters or local neighborhoods is scalable. [4] Clusters with high values of Inter_SS(), can be expected to contain authors or webpages with similar topics and thus merge events that comprise of such clusters are semantically meaningful (Semantic Merges). Hence, the Merge events are ranked in decreasing

[4]Also, note that these operations are performed only on merge and split events detected.

order of the Inter_SS() of the merging clusters. For the Split events, we compute the Inter_SS() of the split clusters. We will illustrate both types of Semantic events in the next section. Note that our toolkit can output Semantic Similarity scores for clusters (not shown).

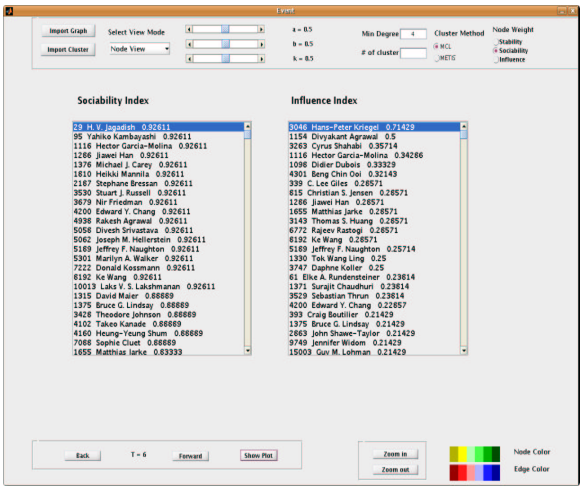


**Figure 4: Node View**

**Node View**– All the above-mentioned views deal with global properties of the network. The node view presents the user with localized information. Once the user chooses node view, the toolkit provides a list of nodes ranked in decreasing order of the properties available (sociability, stability and influence). An example is shown in Fig 4. The user can then select a node from the node list for further observation. This prompts the corresponding neighborhood graphs of that particular node over time to be displayed to the user. The displayed graphs includes the chosen node and its neighborhoods over time. One can gain insight into changes occurring in the neighborhoods of the selected node. For instance, in the case of influence, one can identify spheres of influence for a node over time. We will demonstrate the benefits of the node view in the case studies in the next section.

**Zoom Filters:** As the name suggests, this feature is used to zoom into certain sections of the graph. The user can select the area of interest by drawing a rectangle using the mouse. The selected part is then zoomed into and displayed. It is also possible to zoom out to a lower resolution. Figure 5(a)-(b) demonstrates the zoom feature on the DBLP dataset. In the first figure, the user has selected a part of the graph to zoom into. The second image displays the zoomed part of the graph. The zoom feature can be used multiple times to increase the resolution.

Please note that functionalities provided by node view and the zoom filter are very different. In the node view, the user selects a node and a behavioral characteristic - stability or other metrics, i.e., the user knows which node is of interest to her. However, while using zoom filter, the user must first visually ascertain the part of the graph that is of interest to him, and then seek the details. Explicit node information is not needed for using this filter.

**Time Browser:** This functionality is used to observe the network across time. This provides the user the capability to detect time instants when graph topology has changed considerably. Once the time period is known, the user can use other views to drill down the details. The *Back* and *Forward* buttons at the bottom of the GUI can be used by the user to control the time, moving through the different time intervals.

# 7. CASE STUDIES : VISUAL ANALYSIS

## 7.1 DBLP Bibliography Dataset

In this case study, we demonstrate the effectiveness of our toolkit for visual data analysis on the DBLP dataset. Our tool provides us with a list of authors ranked by behavioral attributes such as Sociability and Influence, as described previously. Our tool also allows one to combine information from multiple metrics by specifying an affine combination of these values (menu-driven option not shown). For this study, we chose Dr Rakesh Agrawal, who unsurprisingly ranks highly on both sociability and influence (see Fig 6). We equally weighted the contribution of each index.

Upon inspection, one can see that the neighborhoods for Dr Agrawal differ significantly between successive snapshots From 1997 to 2002, one can make out the progression in his sociability and influence index, as conveyed by the gradation of the color of the node representing him [5]. After 2002, however, many of his neighbors (collaborators) remain fairly consistent, the sociability index is lower but the influence, of collecting more neighbors in the cluster he is in balances this out very nicely. This correlates with his interests shifting to the focused area of privacy preserving data mining and trust and security applications of databases. His collaborators in the last few timestamps shown are primarily from this area and this area has also taken off very nicely since Agrawal and Srikant's seminal paper in the area in 2001.

Also, one can readily see that 3 nodes in particular appear quite frequently, namely R. Srikant, R. Bayardo and J Kiernan (after 2001) represented by dashed edges. These are some of Agrawal's frequent collaborators. Another interesting trend is that the graph also identifies quite naturally collaborators of Agrawal's who have a high sociability and influence index (i.e. Christos Faloutsos, Gerhard Weikum, Dimitrios Gunopulos, Johannes Gehrke, Prabhakar Raghavan, and Surajit Chaudhary).

## 7.2 Wikipedia webgraph

In the next case study, we analyze the Wikipedia webgraph. In particular, we demonstrate the use of Semantic analysis with event detection. In the event view, the toolkit provides us with a list of different events. As we mentioned in the previous section, we have the facility of ranking the Merge and Split events in terms of semantic meaningfulness. First, we will consider a *Semantic Merge* event, that had the highest Semantic Similarity, shown in in Figure 7(a). In the first snapshot, there are 2 clusters of size 5 and 11, that have considerable semantic similarity, as can be ascertained from the labels. The clusters deal with *Logic* and *Philosophy* and the Merge event is thus justifiably meaningful. The merged cluster is shown on the right.

For a Split event, one would expect the two split clusters to be semantically dissimilar. While, this is mostly true, there can be occurrences where interesting minute differences can cause clusters to split up. These kind of Split events can in-

[5]In 1997 this value is low simply because this is the first data point in the dataset we use – an artifact of the experiment.

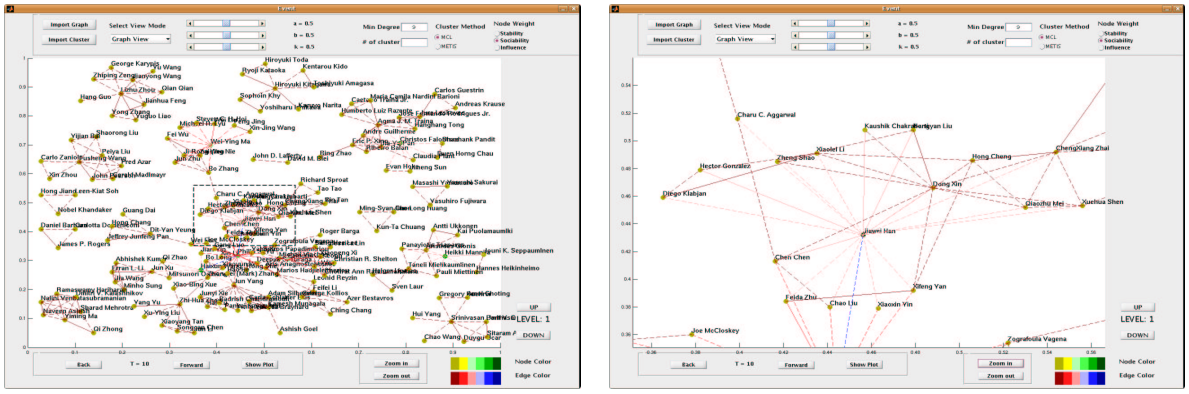

**Figure 5: Zoom feature**

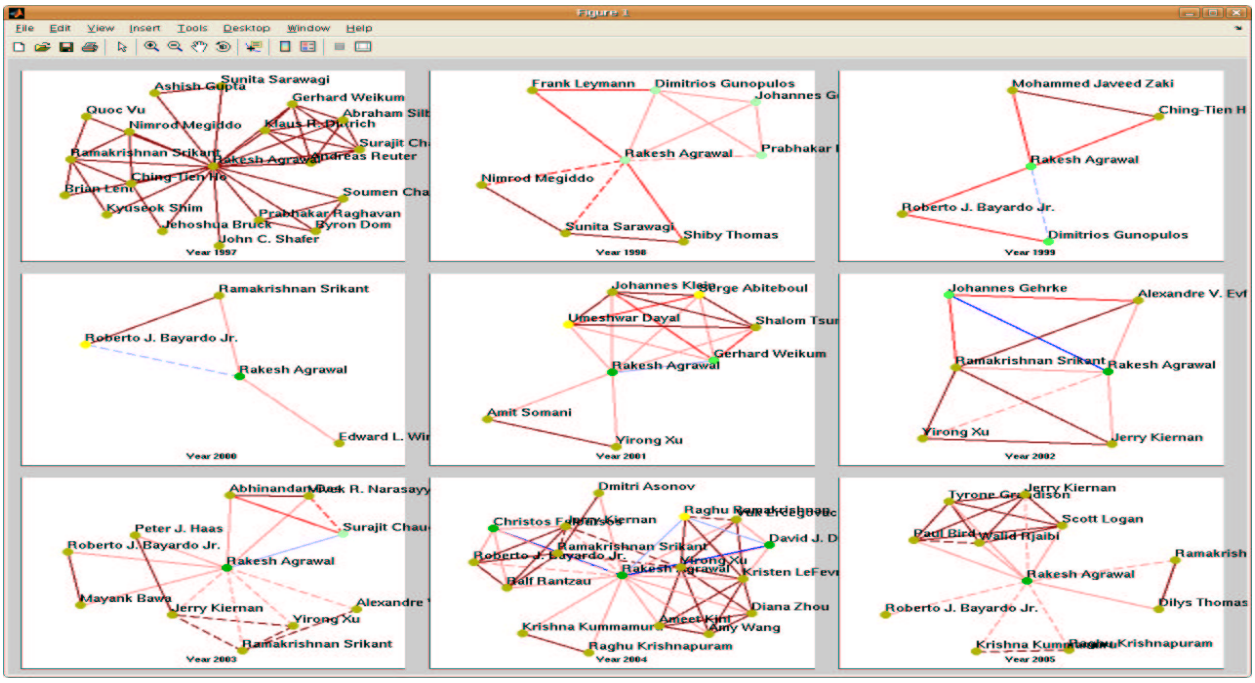


**Figure 6: Node View : Neighborhood of R. Agrawal (weighted by Sociability Influence Index)**

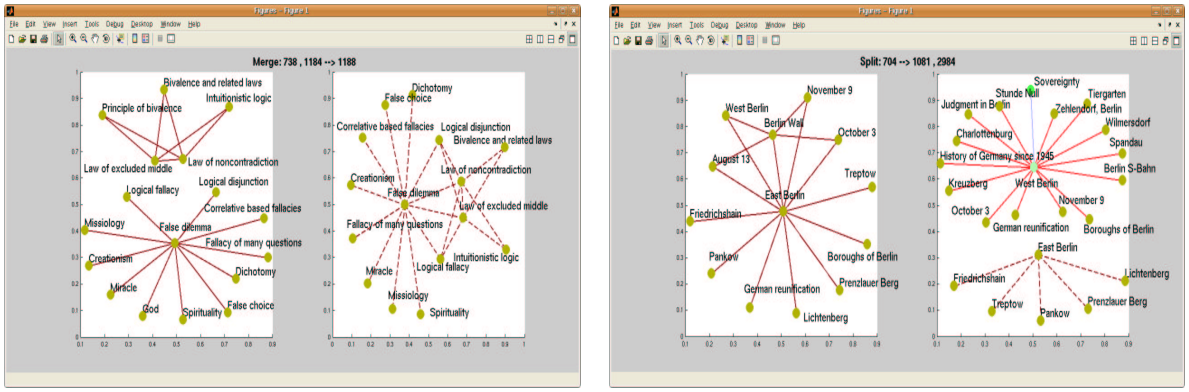


**Figure 7: Examples of a) Merge Event b) Split Event**

dicate subtle changes across snapshots, where a cluster splits into two parts due to a small semantic difference among the associated categories. These splits can be interesting as they can reveal differences that may not be obvious. This can be considered akin to drilling down a hierarchy to discover subtle specializations of a category. We can find such interesting occurrences by considering split clusters with low Inter_SS(). An example of such a Split event is shown in Figure 7(b). As can be observed, the original cluster which deals with the Berlin Wall, splits into two clusters on East and West Berlin respectively. Understandably, these two split clusters had reasonably high Semantic Similarity values.

# 8. CONCLUSIONS AND FUTURE WORK

We have presented a toolkit for visualizing and analyzing dynamic interaction graphs. Our toolkit provides multiple views of the data and is designed to incorporate features for multiscale and multi-resolution analysis and supports the *overview, zoom, filter and details–on–demand paradigm.* The toolkit also supports visualizing the cumulative graphs with different scoring mechanisms to assign color to each edge and node. The coloring method captures behavioral properties and provides useful visual cues to discover important and interesting parts of the graph. We have shown how the toolkit can perform visual analysis by taking into account the evolution of nodes and communities as well as key events over time. Using illustrations on the DBLP and Wikipedia datasets, we have shown how the interactive features aid the user in answering common queries about dynamic networks in effective and efficient manner. In the future, we would like enhance the toolkit with additional features and further improve the scalability and performance.

# 9. ACKNOWLEDGEMENTS

This work is supported in part by the DOE Early Career Principal Investigator Award No. DE-FG02-04ER25611, NSF CAREER Grant IIS-0347662 and NSF SGER Grant IIS-0742999. We would like to thank Klaus Berberich and Evgeniy Gabrilovich for providing us with the Wikipedia dataset and the category hierarchy respectively.

# 10. REFERENCES

[1] J. Abello, F. van Ham, and N. Krishnan. Ask-graphview: A large scale graph visualization system. *IEEE Trans. Vis. Comput. Graph.*, 12(5):669–676, 2006.

[2] S. Asur, S. Parthasarathy, and D. Ucar. An event-based framework for characterizing the evolutionary behavior of interaction graphs. *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 913–921, 2007.

[3] L. Backstrom, D. P. Huttenlocher, and J. M. Kleinberg. Group formation in large social networks: membership, growth, and evolution. *SIGKDD*, 2006.

[4] K. Berberich, S. Bedathur, T. Neumann, and G. Weikum. A time machine for text search. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 519–526, New York, NY, USA, 2007. ACM.

[5] S. K. Card, J. D. Mackinlay, and B. Shneiderman. *Readings in information visualization: using vision to think*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[6] D. Chakrabarti, R. Kumar, and A. Tomkins. Evolutionary clustering. *SIGKDD*, 2006.

[7] T. Falkowski, J. Bartelheimer, and M. Spiliopoulou. Mining and visualizing the evolution of subgroups in social networks. *IEEE/WIC/ACM International Conference on Web Intelligence*, 0, 2006.

[8] P. A. Gloor, R. Laubacher, S. B. C. Dynes, and Y. Zhao. Visualization of communication patterns in collaborative innovation networks - analysis of some w3c working groups. In *CIKM*, pages 56–60, 2003.

[9] P. A. Gloor and Y. Zhao. Analyzing actors and their discussion topics by semantic social network analysis. In *IV*, pages 130–135, 2006.

[10] J. Heer and D. Boyd. Vizster: Visualizing online social networks. In *INFOVIS*, page 5, 2005.

[11] N. Henry and J.-D. Fekete. Matrixexplorer: a dual-representation system to explore social networks. *IEEE Trans. Vis. Comput. Graph.*, 12(5):677–684, 2006.

[12] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. *SIGKDD*, 2003.

[13] G. Kumar and M. Garland. Visual exploration of complex time-varying graphs. *IEEE Trans. Vis. Comput. Graph.*, 12(5):805–812, 2006.

[14] J. Leskovec, J. M. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. *SIGKDD*, 2005.

[15] M. Newman. Clustering and preferential attachment in growing networks. *Phys. Rev. E*, 64, 2001.

[16] A. Perer and B. Shneiderman. Balancing systematic and flexible exploration of social networks. *IEEE Trans. Vis. Comput. Graph.*, 12(5):693–700, 2006.

[17] E. Qeli, W. Wiechert, and B. Freisleben. Visual exploration of time-varying matrices. In *IV*, pages 889–895, 2005.

[18] P. Resnik. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of Artifical Intelligence Research*, 11:95–130, 1999.

[19] R. Samtaney, D. Silver, N. Zabusky, and J. Cao. Visualizing features and tracking their evolution. *IEEE Computer*, 27(7):20–27, 1994.

[20] M. Spiliopoulou, I. Ntoutsi, Y. Theodoridis, and R. Schult/. Monic - modeling and monitoring cluster transitions. *SIGKDD*, 2006.

[21] J. Sun, C. Faloutsos, S. Papadimitriou, and P. S. Yu. Graphscope: parameter-free mining of large time-evolving graphs. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 687–696, New York, NY, USA, 2007. ACM.

[22] H. Yang, S. Parthasarathy, and S. Mehta. Mining spatial object patterns in scientific data. *Proc. 9th Intl. Joint Conf. on Artificial Intelligence*, 2005.