On the Use of Viewpoint Neighborhoods for Dynamic Graph Analysis

Sitaram Asur and Srinivasan Parthasarathy #

Department of Computer Science, Ohio State University
Columbus, OH 43210, U.S.A
#Contact Author
(asur,srini)@cse.ohio-state.edu

ABSTRACT

Recent innovations have resulted in a plethora of social applications on the Web, such as blogs, social networks, and community photo and video sharing applications. Such applications can typically be represented as evolving interaction graphs with nodes denoting entities and edges representing their interactions. The study of entities and communities and how they evolve in such large dynamic graphs is both important and challenging.

While much of the past work in this area has focused on static analysis, more recently researchers have investigated dynamic analysis. In this paper, we focus on dynamic graph analysis, but in a departure from recent efforts, we consider the problem of analyzing patterns and critical events that affect the dynamic graph from the *viewpoint* of a single node, or from the *viewpoint* of a selected subset of nodes. Defining and extracting a relevant viewpoint neighborhood efficiently, while also quantifying the key relationships among nodes involved are the key challenges we address. We also examine the evolution of viewpoint neighborhoods for different entities over time to identify key structural and behavioral transformations that occur. We illustrate the benefits of such an analysis on two real-world graphs - the DBLP co-authorship network and the Wikipedia webgraph.

1. INTRODUCTION

The trend in online computing has shifted to a social context, with social communities such as Facebook, MySpace and Orkut, weblogs and community photo and video sharing applications gaining tremendous popularity. These online social networks share common attributes with other real-world networks such as co-authorship networks and the Web, in that, they can be efficiently represented as an interaction graph, where nodes denote entities of interest, and the ties among entities can be modeled as edges. The study of such interaction networks can provide insight into the structure and function of such systems ([4, 3, 17]), potentially allowing one to predict interesting aspects of their behavior. Such analysis is critical to online applications such as search ([2, 20, 14]) and advertising ([1, 15]).

An important challenge is that these networks are usually evolving in nature, with the addition and deletion of edges and nodes representing changes in the interactions among the modeled entities. Characterizing and modeling the changes that occur can go a long way in aiding the development of generic models for understanding and reasoning about these evolving networks ([4, 3, 24, 17, 19]).

An important aspect of analyzing these graphs is to understand the dynamic behavior of nodes over time. The changing interactions of nodes, and their influence on other nodes ([10, 16, 3]), can assist in inferring future interactions as well as predicting trends in community evolution. Also, one may be interested in analyzing or reasoning about how *the connectivity of the interaction graph* changes over time from the view point of a single node, a group of nodes or a community. This enables one to study the isolated effects caused by global changes to particular nodes in the graph.

In the setting of targeted advertising on an on-line social network like Facebook, advertisers targeting an individual can glean useful information regarding the propensity of the person being responsive to a product by studying the local neighborhood of that individual, the presence of influential members in the immediate friend- circle as well as the nature of the relationships among them. The key challenge is to identify and quantify these relationships for different nodes based on the topology of the graph. Also, most online communities provide facilities for storing content apart from link information [2]. We would like to consider how content information such as semantic features can be incorporated along with link information to capture relationships among nodes. In the first half of the paper, we concern ourselves with the following important question. *How can we identify the local neighborhood of interest for a particular source node, and also quantify the impact or importance of different nodes in the constructed neighborhood, with regard to the source node?*

To obtain a satisfactory answer to this question, we formally define the notion of a *Viewpoint Neighborhood* of a node, which represents the immediate neighborhood of interest for a particular node. We then discuss the properties that need to be modeled to measure importance and effect, and arrive at an activation spread model for identifying the members of a node's immediate neighborhood. The proposed model uses an activation function to select nodes of interest with respect to a particular source node, and also to quantify the level of their involvement using commitment values. We show how different activation functions can be employed capturing different intrinsic and extrinsic properties of nodes in the graph. We also extend this problem to one where we wish to identify the common shared neighborhood for a set of nodes, which is important in keyword search and influence maximization applications.

WWW2009, April 20-24, 2009, Madrid, Spain.
.

Since the graphs are evolving, the neighborhoods as well as the relationships will change over time. A crucial problem in this context is to characterize the changes occurring in these neighborhoods over time and show how they can be used to build models for dynamic behavior. For this purpose, we define certain temporal events that can characterize changes and measure important structural and behavioral patterns such as stability and popularity over time.

Also, changes typically impact different entities in the graph in different ways. *How can one identify important changes that have a large effect on the nodes of the graph?* We show how the effect of changes on specific nodes in the graph can be gleaned by temporal analysis on the Viewpoint Neighborhoods for nodes in the evolving graph. In this regard, we introduce *core subgraphs*, which represent stable sections of a node's neighborhood, and $transformation$ *subgraphs* which represent changing sections of the neighborhood. We show how frequent subgraph mining can be leveraged to capture these motifs. To provide illustrations, we make use of two real-world interaction graphs - the DBLP co-authorship network and the Wikipedia webgraph.

To sum up, the key contributions of this work include:

- The description and formalization of Viewpoint Neighborhoods to represent the neighborhood of interest for a node.
- A general activation model for identifying Viewpoint Neighborhoods for a node. Presentation of different activation functions to capture topological and semantic properties.
- Extension to the multi-source case, with an algorithm to identify the shared Viewpoint Neighborhood of a group of query nodes.
- The identification of key temporal events and incremental behavioral measures to characterize changes that occur in neighborhoods over time.
- Introduction of core subgraphs and transformation subgraphs to capture different effects of changes to the neighborhoods of nodes over time.

The paper is organized as follows. We will describe some related work in the next section. In Section 3, we will outline the details of the datasets we employ in this work. In Section 4, we provide a formal definition of Viewpoint Neighborhoods and the Activation spread model. In Section 5, we describe temporal analysis on the Viewpoint Neighborhoods and the temporal events and behavioral measures in this context. We conclude in Section 6.

2. RELATED WORK

Recently, there has been considerable interest in analyzing dynamic interaction graphs. Leskovec *et al* [19] 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. In later work, Leskovec and others [18] studied individual node arrival and edge creation processes that collectively lead to macroscopic properties of dynamic social networks. In particular, they investigated the role of edge locality in network evolution. Kumar and others [17] have analyzed the evolution of structure in social networks, providing measurements on two real-world networks. Backstrom *et al* [4] have focused on the formation of groups and the ways they grow and evolve over time. Yahia and others [2] have surveyed search tasks on social communities and discussed relevance measures for efficient searching and ranking of social content. Tantipathanandh and others [25] have developed a framework for detecting dynamic community structure in evolving graphs. They make use of dynamic programming and heuristics to optimize the structure discovered. Sun and others [24] have proposed GraphScope for parameter-free pattern mining of time-evolving graphs. Longitudinal analysis of time-varying networks has also attracted some attention. Snijders has a large body of work on inferential statistics for longitudinal network analysis focusing particularly on actor-oriented modeling of dynamic data ([23]).

Event-based methods have been applied for target tracking [22] and studying the evolution of spatial data [27]. In our earlier work [3], we proposed an event-based framework for characterizing evolving interaction graphs. In this regard, we examined clusters of graphs and outlined events for their changes over time. However, clusters are dependent on the clustering algorithm applied and are restricted by the fact that they do not capture local relationships. All the nodes within a cluster are treated the same, without any structure information. This makes it infeasible to study local relationships and their evolution over time, which is the main aspect of this work. Here, we focus on identifying neighborhoods making use of topological and semantic information, while retaining local structural information. This enables us to examine and quantifying relationships within local neighborhoods and how they evolve over time.

Faloutsos and others [11] have examined an electric current based approach to identify connection subgraphs. Also, Tong and Faloutsos [26] have presented the notion of center-piece subgraphs, where the goal is to identify a small but representative connection subgraph, given a set of query nodes. The Center-piece subgraph for a set of nodes represents the important nodes with close connections to all or some of them, similar to the notion of centrality. The authors have used random walks with restart from the query nodes to identify important nodes using a path-based goodness score function. Although, this is similar to the multi-source neighborhoods that we discuss in Section 4.5, a key difference is that our algorithm makes use of additional constructs such as local topological properties as well as semantic information to construct neighborhoods. Also, our activation spread algorithm is general, in that, activation functions making use of different properties or even a combination of properties can be employed to extract neighborhoods efficiently.

Activation models have been studied for complex networks specifically in the context of influence maximization ([15, 16, 7, 1, 9]). Kempe et al ([15, 16]) discuss two models for the spread of influence through social networks. Their model however is computed by simulation on a static network. It does not consider the addition and deletion of nodes and edges in the network. There has also been research on using activation functions in keyword search.([20, 14, 7]). In this work, we develop an activation model for the important problem of identifying neighborhoods for nodes in dynamic graphs and quantifying relationships within them as well as their evolution over time. For this purpose, we introduce a general model along with three different activation functions.

3. DATASETS

We use two different real-world interaction graphs for our

experimental analysis.

DBLP co-authorship network: We used the DBLP computer science bibliography data to generate a co-authorship network representing authors publishing in several important conferences in the field of databases, data mining and AI. We chose all papers over a 10 year period (1997-2006) that appeared in 28 key conferences spanning mainly these three areas. The conferences we considered are - (*PKDD, ACL, UAI, NIPS, KR, KDD, ICML, ICCV, IJCAI, CVPR, AAAI, ER, COOPIS, SSDBM, DOOD, SSD, FODO, DASFAA, DEXA, ICDM, IDEAS, CIKM, EDBT, ICDT, ICDE, VLDB, PODS, SIGMOD*). 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 [1]. The graph spanning 10 years contained 23136 nodes and 54989 edges. We chose the snapshot interval to be a year, resulting in 10 consecutive snapshot graphs. As researchers have noted ([15, 5]), a collaboration network exhibits many of the structural properties of large social networks and is hence a good representative dataset for this analysis.

Wikipedia Revision History 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 [6]. 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. To perform semantic analysis,we make use of a category hierarchy, which we have obtained from Gabrilovich [13]. 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 10 snapshots for our analysis.

4. VIEWPOINT NEIGHBORHOODS

In this section, we introduce the notion of Viewpoint Neighborhoods and discuss algorithms to find them.

4.1 Problem Definition

Our goal here is to identify a neighborhood of interest for a given source node as well as quantify the relationship or effect of different nodes in the neighborhood on the source node.

Definition: We can define a *Viewpoint neighborhood* (VPN) for a given source node x as the graph (or a tree in the special case) rooted at x containing only nodes with some degree of importance to x and their interconnections.

The above definition refers to nodes of importance to the given source node. Our goal in this section is to devise an efficient algorithm for identifying these nodes that make up the neighborhood of interest, and quantify their relative importance. We begin by presenting a simple algorithm based on distance, and then discuss its shortcomings, which leads us to a better and more efficient algorithm.

[1]Note that edges are unweighted i.e if two authors had at least one publication, they would be represented by an edge.

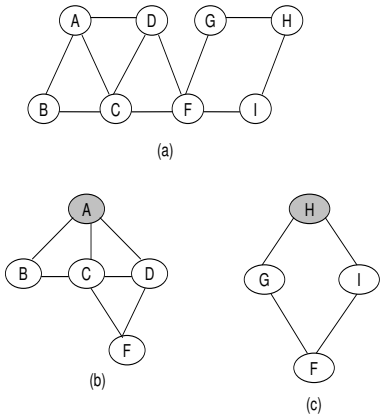


Figure 1: Example of a k-viewpoint neighborhood. (a) represents the original graph. (b) corresponds to 2-viewpoint of node A. (c) represents 2-viewpoint neighborhood of node H

4.2 Depth-limited VPN

Consider a social network where nodes represent people and links represent the friendship ties among them. Since the network captures real-world behavior, a node's real-life friends are likely to be linked either directly to the node or within two hops. Since nodes that are closer to x are likely to have a greater effect on x than nodes further away, our initial attempt at an algorithm would be to consider all nodes within a particular distance as part of the neighborhood, formally defined as follows.

Definition: Let dist(a,b) represent the shortest distance in hops from node a to node b. A k-viewpoint neighborhood of node i for a graph G=(V,E), is a subgraph consisting of only vertices, $V' \in V$ such that $\forall v \in V', dist(v,i) <= k$, and all edges $E' \in E$ such that $\forall (v_i, v_j) \in E', v_i \in V'$ *and* $v_j \in V'$.

Algorithm 1 Find-kVPN(G,src,k)

```
Input: Graph G = (V, E), k, the size of the neighborhood
required and src, the source node.
Mark src as visited
if depth + 1 ≤ k then
  for each neighbor j of node src do
    Add edge (src, j) to VPN
    if j has not been visited then
      Find-kVPN(G,j,k)
    end if
  end for
end if
return(VPN)
```

For example, consider the graph in Figure 1 (a) consisting of 8 nodes and their connections. Figure 1 (b) and (c) represent the 2-viewpoint neighborhoods of nodes A and H respectively. Note that the structure and members of the two neighborhoods differ greatly with only node F in common. A k-Viewpoint neighborhood for a node can be computed in a straightforward manner by performing depth-limited search from that node, as shown in Algorithm 1. Given a

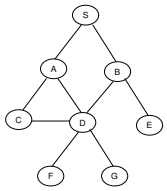


Figure 2: In this example, nodes D and E are at the same distance from the query node S but their link structure differs.

value of k, the traversal is carried out until the required depth is attained. However, the above definition is relatively naive since it makes the assumption that all nodes are considered equal in terms of their involvement in the neighborhood, and in particular all nodes within a particular distance k from the source node will all belong to the k-VPN. This assumption is not true, as two nodes that are the same depth away, may not necessarily impact the source node in the same way. One of them might be well-connected and hence linked to many other nodes within the neighborhood, while the other might be a singleton, with a degree of 1. An example is shown in Fig 2. We can see that nodes D and E are at the same distance from source node S but while D is well-connected with other nodes in S's neighborhood, E is connected to just one node. We can justly argue that D and E impact node x differently and this needs to be implicitly captured by the algorithm. Hence, while finding the VPN for a node, we need to not only consider whether or not a node belongs to a neighborhood, but also its degree of commitment towards that particular neighborhood. Note that, this is not an edge weight. Instead, we associate a commitment or importance value for a node with respect to each neighborhood (VPN) in its environs. Depending on the application, the measure for computing this quantity can vary.

Also, given the above intuition, it is clear that, to identify nodes that are important to a particular node in a social network, we need to concentrate not only on distance from the source node but also *local connectivity* information.

A third thing to consider in this context is that, in small-world graphs, hub nodes have interactions with most nodes in the graph. This makes path lengths from one end of the graph to another very small. Hence, even for small values of k, the VPN for a node might include a large portion of the graph, which is not likely to be useful. Not only are the neighborhoods going to be unnecessarily large to store, but it severely impacts the analysis that can be done over them. A close friend who is connected to a lot of people can be important to a node, but if a friend is connected to a hub, who is in turn connected to a bunch of people whom the source does not know, then that hub is not likely to be important. We need to differentiate these cases.

4.3 Activation Spread

Considering the discussion presented above, the algorithm for constructing the Viewpoint neighborhood of a node needs to satisfy three important criteria.

- **Inverse Distance weighting** : The probability of involvement of a node to a neighborhood should be inversely proportional to its distance from the source. The intuition is that a node is likely to be affected more by changes occurring near itself than those occurring some distance away.
- **Link Structure** : Nodes that are well-connected ,i.e having links to many other nodes within the neighborhood, should have high commitment or importance values. These are nodes that have high influence within the neighborhood.
- **Hub nodes** : As mentioned above, hub nodes distort neighborhoods by bringing 'uninvited guests' - a host of other nodes that do not belong. The algorithm should expand such hub nodes with low probability.

We propose a general activation spread model which is designed to satisfy the above three criteria, and construct a VPN for a given source node. To simplify matters, we consider undirected graphs in this paper, although the algorithms do not make any such assumptions. The general model is as follows. Activation begins at the source node. We assume that a budget M is initially available. The source node distributes this amount among its immediate neighbors, initiating the activation process. Each node then retains some amount for itself and splits the remainder among its other neighbors (siblings and descendants). By descendants, we mean all nodes that are farther away (have greater distance in terms of hops) from the source node than the current node. Thus a node is expanded only once. If a node has already been activated, it is not expanded a second time. To handle the inverse-weighting of nodes, we decay the activation as it proceeds farther away from the source node. Each time the activation touches a node, it decays by a factor of the number of the links the node has. This ensures that nodes closer to the source node are more probable to be chosen. When the spread reaches a node, the node is chosen to belong in the VPN based on certain criteria such as its degree or topology or even semantic features. A node that is well-connected will thus benefit from having multiple connections. Key to the effectiveness of the spread is *an activation function* that serves as a distribution mechanism. At each node, the activation function determines the amount assigned to the node. We will discuss different activation functions later in this section. The spread algorithm is presented as Algorithm 2.

The activation proceeds with the amount constantly decaying until reaching a minimum threshold, at which it is deemed indivisible. At the end of the activation process, the total of the amounts at each node sums to M. Hence, the fraction of the total amount that each node has received gives the commitment value for a node in this particular VPN. Thus the importance of a node is proportional to its local connectivity and also depends on its path from the source node.

THEOREM 4.1. *The spread algorithm outlined above satisfies the three criteria provided earlier.*

PROOF. During activation spread, the amount transmitted by the source is constantly decayed as it moves further and further away, since each node retains a portion (> 0) and distributes the rest, satisfying the inverse distance weighting criterion. Also it will satisfy condition 2 since common-neighbors of nodes will receive portions from each of the nodes, thereby ending up with higher amounts than nodes that are connected to only one of the earlier nodes. Finally, hubs will have low importance since a node retains only one portion after dividing among all its neighbors. So if a node has a large number of neighbors it will

Algorithm 2 Find-VPN(*Adjlist*, *src*, M, *min_thresh*)

Input: Adjacency list *Adjlist* , *src*, the
start node, M, the budget and *min_thresh* the stopping threshold
Output: Commitment P
Initialize commitment values $P(x) = 0$ for all nodes
for each neighbor y of source *src* **do**
  /* Push the node along with the amount it needs to receive into queue */
  Push (y,$Act(y, M)$)
end for
while queue is not empty **do**
  Pop node-amount pair from the queue as (x,m)
  if x has already been expanded or $\frac{m}{deg(x)} < min\_thresh$ **then**
    Add amount $Act(x, m)$ to $P(x)$
    Continue
  end if
  /* Expand node x */
  for each descendant or sibling neighbor y of x **do**
    if $Act(y, m) < min\_thresh$ **then**
      /* No need to expand that node */
      Add amount $Act(y, m)$ to $P(x)$
    else
      /* Enqueue for activation */
      if y is already on the queue and its predecessor is the same level as x **then**
        Add amount $Act(y, m)$ to what y is going to receive
      else
        Push (y,Act(y, m)) into the queue
      end if
    end if
  end for
  Mark x as expanded
end while
for each expanded node **do**
  $P(x) = \frac{P(x)}{M}$
end for
Return P

split up what it has received over all of them and will be left with a very small portion.

COROLLARY 4.2. *The activation spread algorithm converges in finite time due to the perpetual decay of the amount being propagated* [2].

Note that, in the above algorithm we considered each edge to be the same. If edge weights are available, then during the activation propagation, a node can consider edge weights while dividing the amount among its neighbors. Each neighbor node will not receive the same share in that case.

4.4 Activation Functions

The activation function is used by the spread algorithm to perform the distribution of different amounts to different nodes depending on topological or semantic features. We will next present three different activation functions, the first based on inverse-degree, the second based on local Betweenness Centrality and the third based on semantic content. Note that, it is possible to design a function to incorporate multiple features.

Inverse Degree Activation: This is a simple activation function that down-weights nodes with high degrees. Let *downlinks*(x) represent the siblings and descendants of a node x. When each node x, except the source node, receives some amount, it retains $\frac{1}{|downlinks(x)|+1}$ and distributes the same fraction to each of its siblings and descendants. Note that, hub nodes that are connected to a large number of

[2]We are using a threshold to hasten the convergence

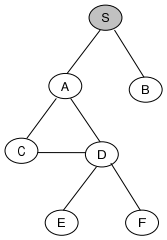


Figure 3: In this example, using Inverse Degree activation, node B is going to get higher weight although it is poorly connected.

nodes, will retain small amounts using this function.

$$Act(x, m) = InvDeg(x, m) = \frac{m}{|downlinks(x)| + 1}$$

This activation function satisfies the three criteria we discussed previously, with inverse weighting of nodes taken care of by the decay, common neighbors getting higher weights and hubs down-weighted. However, it does not completely capture the link structure in the graph. We can illustrate this with an example, shown in Fig 3. Nodes A and B are both going to receive $\frac{M}{2}$ from the source node. While A has 2 downlinks, and hence retains only $\frac{M}{6}$, node B will retain the $\frac{M}{2}$ it received since it does not have any descendants. Node A is connected to all other nodes in the neighborhood and should receive better recognition than it does using this activation function. Hence, the activation function at a node needs to consider not only immediate links but also more global topological information. We will show how this can be improved with the use of the Betweenness topological measure next.

Betweenness-based Activation: The Shortest-path Edge betweenness centrality measure, captures an important topological property, and was first introduced by Freeman[12]. It is a global topological measure and computes, for each node in the graph, the fraction of shortest paths that pass through it. In our case, we are interested in the reachability of nodes in the neighborhood from a given source node. Hence, we need to favor nodes that have high Shortest-path Edge Betweenness in terms of paths from the source to other nodes in the neighborhood. These nodes have high centrality within the VPN and thus can be considered important nodes.

The Shortest-path Edge Betweenness centrality for a given node x with respect to a source node S given a neighborhood of nodes $N = (V, E)$ can be calculated as :

$$B(x, S) = \frac{SP_x}{|V| - 1} \quad (1)$$

where SP_x is the number of shortest paths passing through edge x from source S to the nodes of the neighborhood V.

The shortest-path edge betweenness centrality can be computed by performing BFS from the source node, building the shortest path tree for nodes in the neighborhood. Since we are considering local neighborhoods, we are interested only in betweenness centrality within the VPN. This quantity is less expensive to compute than the global betweenness centrality in the graph. However, we need to know

the key members of the VPN before computing betweenness. For this, we simulate an activation spread, using the Inverse-degree activation function and simultaneously compute shortest paths. The algorithm *FindBet* is shown in Alg 3.

Once the betweenness values are obtained for nodes in the neighborhood, they can be used in the activation function. When a node receives an amount, it evaluates its own betweenness and the betweenness values of its siblings and descendants, and distributes the amount as a ratio of these values. Let the amount received by a node x be m. Let the sum of the betweenness of its siblings and descendants be denoted as $B(downlinks_x)$. The amount retained by node x is given as:

$$Act(x,m) = \frac{B(x) * m}{B(downlinks_x) + B(x)}$$

The amount received by each of its siblings and descendants y is given similarly as :

$$Act(y,m) = \frac{B(y) * m}{B(downlinks_x) + B(x)}$$

In the example shown in Fig 3, A will receive a larger amount ($\frac{5M}{6}$) than B ($\frac{M}{6}$) from S.

Algorithm 3 Find-Bet(*Adjlist*, *src*, *M*, *min_thresh*)

Input: Adjacency list *Adjlist* , *src*, the start node, M, the budget, Activation Function $InvDeg$ and min_thresh the stopping threshold
Output: Betweenness B
Initialize depths $D(x) = Inf$ for all nodes
Initialize predecessor list Pr
for each neighbor y of source src **do**
  Push (y,$InvDeg(y,M)$,1,src) into queue
end for
while queue is not empty **do**
  Pop node-amount-depth-predecessor tuple from the queue as (x,m,d,p)
  if $D(x) = d$ **then**
    /* Alternate predecessor to x along the shortest path from the source src */
    Add predecessor p to $Pr(x)$
  else if $D(x) > d$ **then**
    /* Found shorter path to x */
    Clear predecessor list $Pr(x)$
    Add p to $Pr(x)$
    Set $D(x)$ to d
  end if
  if $InvDeg(x,m) < min_thresh$ **then**
    Do not expand node x
  else
    /* Expand node x */
    for each descendant or sibling neighbor y of x **do**
      if $D[y] < D[x] + 1$ and previous maximum amount received by $y > InvDeg(x,m)$ **then**
        /* No need to expand that node */
      else
        /* Enqueue for activation */
        Push (y,$InvDeg(y,m)$,$D[x]+1$,x) into the queue
      end if
    end for
  end if
end while
for each expanded node x **do**
  Use predecessor list Pr to find path to source and update betweenness B of nodes along the way
end for
Return B

Semantic Activation: Apart from topological features, it is possible to incorporate semantic properties of nodes to encode the activation function. This is of particular importance in personalized and keyword search applications ([2, 20, 14]), where one is interested in identifying subgraphs that match given sets of keywords [20]. To obtain efficient local neighborhoods when nodes are annotated with semantic terms, we need to consider the similarity of nodes with the source node. Let us consider two nodes x and y each associated with sets of terms denoted as K^x and K^y where $K^x = \{k_1^x, k_2^x, ..., k_{|K^x|}^x\}$. A simple way to compute similarity for the two nodes would be to consider the Jacquard measure of their term sets.

$$SS(x,y) = \frac{|K^x \cap K^y|}{|K^x \cup K^y|}$$

However, the relationship between two nodes cannot typically be inferred by merely comparing their term sets since different terms are associated with different semantic meanings. One needs to consider the distribution of topics and the relationships among them. When the terms are organized in a category hierarchy, we can use the notion of semantic similarity to serve our purpose. To begin with, the Information Content (IC) of a term (category or keyword-set), using Resnik's definition [21], is given as:

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

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:

$$SS(k_i,k_j) = IC(lcs(k_i,k_j))$$

where $lcs(k_i,k_j)$ refers to the lowest common subsumer of terms k_i and k_j. The semantic similarity between the two nodes can be formulated as follows :

$$SS(x,y) = \sum_{i=1}^{|K^x|}\sum_{j=1}^{|K^y|} SS(k_i^x, k_j^y) \quad (2)$$

While performing activation spread, we are interested in the semantic similarity between the source node and all other nodes in its neighborhood. While distributing amounts among nodes, we need to provide higher preference to nodes that are semantically more similar with the source node. Let s denote the source node. Let the amount received by a node x be m, and the sum of the semantic similarity of each of its siblings and descendants with the source node be $S(downlinks_x) = \sum_{i \in downlinks_x} SS(s,i)$. The amount retained by node x can then be given as:

$$Act(x,m) = \frac{SS(s,x) * m}{S(downlinks_x) + SS(s,x)}$$

Similarly, the amount received by each of its siblings and descendants, denoted as y, is computed as:

$$Act(y,m) = \frac{SS(s,y) * m}{S(downlinks_x) + SS(s,x)}$$

We present an example of a VPN using semantic activation on the Wikipedia webgraph in Fig 4. The source

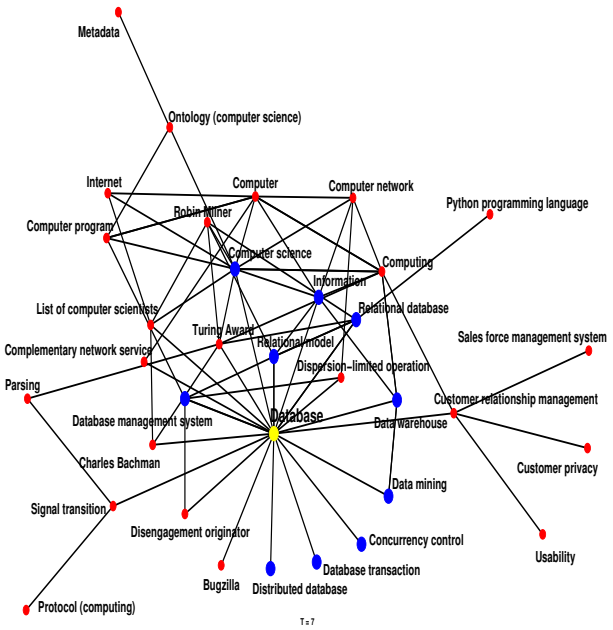


Figure 4: Example of a semantic viewpoint neighborhood. The source node "Database" is shown in yellow. The relatively important nodes are shown in blue.

node is "Database" (shown in yellow), and the nodes with high importance (> 0.025) are shown in blue. The node "Database" had a high degree in that particular snapshot, and was connected to a host of extremely unrelated nodes in the Wikipedia webgraph, such as "Communications in Israel", "430 BC", "Denver-Aurora metropolitan area" and "The Chicago Manual of Style". This is due to spurious links among webpages. Making use of the semantic similarity can help extract a relevant neighborhood for a node, which is extremely important in search ([2, 20, 14]) and spam applications. Note, that it is also possible to find the neighborhood for a node based on a particular input keyword-set. In that case, the activation function needs to consider the semantic similarity of nodes with the source as well as with the query keywords.

As we mentioned previously, it is possible to construct an activation function to consider both the betweenness as well as the semantic features, using weights to tune the relative importance of each property. Figure 5 gives the distribution of the VPNs obtained for the DBLP and Wikipedia datasets using the Betweenness-based and Semantic Activation functions respectively. We can observe that for DBLP, the neighborhood size distributions across time are close and overlapping. In the case of Wikipedia, the number of nodes increases constantly across timestamps, which causes the sizes of the neighborhoods and their number to increase, but the distribution pattern remains similar.

4.5 Multi-source Neighborhoods

Note, that our definition can be extended to the multi-source case, where we are interested in identifying the shared central neighborhoods given multiple source nodes. The problem can be stated as follows.

Problem : Given a set of n source nodes $S = (s_1, s_2, ..., s_n)$, the problem is to find the VPN that represents the intersection of at least k of their neighborhoods.

This is important again in keyword search applications, where one wishes to compute a subgraph to satisfy a given keyword-set. The nodes that have high importance in this shared neighborhood should represent those that have involvement with the VPNs of at least k of the source nodes. On the other hand, nodes that occur in only a few of the VPNs should have 0 importance. We would like to design

Algorithm 4 MultiVPN($Adjlist$, $srclist$, M, min_thresh, k)

Input: Adjacency list $Adjlist$, $srclist$, the list of source nodes, M the budget, min_thresh the stopping threshold and k the minimum number of VPNs the node should belong to
Output: Commitment Values P
/* Find individual neighborhoods for the source nodes as described previously */
for each source node src **do**
  $BTemp_{src}$ = Find-Bet($Adjlist$, src, M, min_thresh)
  $PTemp_{src}$ = Find-VPN($Adjlist$, src, M, min_thresh)
end for
/* Coalesce individual betweenness and commitment values to obtain betweenness and importance with regard to the different source VPNs */
for each node x **do**
  $Bet_{all}(x) = \sqrt{\sum_{i=1}^{|srclist|} BTemp_i(x)^2}$
  /* Prune all nodes that do not belong to at least k out of $|srclist|$ neighborhoods */
  if x belongs to at least k out of $|srclist|$ VPNs **then**
    $P_{all}(x) = \sqrt{|srclist|} - \sqrt{\sum_{i=1}^{|srclist|} (1 - PTemp_i(x)^2)}$
  else
    $P_{all}(x) = 0$
  end if
end for
Initialize final commitment values $P(x) = 0$ for all nodes
for each node y that belongs to at least k out of $|srclist|$ neighborhoods **do**
  /* Begin activating from the node with amount proportional to its importance wrt. all source nodes */
  Amt $= \frac{P_{all}(y) * M}{\sum_{i \in nodelist} P_{all}(i)}$
  P_y = Find-VPN($Adjlist$, $srclist$, Amt, min_thresh)
  Update commitment values of nodes with P_y
end for
for each node x **do**
  $P(x) = \frac{P(x)}{M}$
end for
Return P

an activation model that results in the nodes well connected to the neighborhoods of the source nodes having high centrality and thus high weights. Such an activation model would proceed in a different vein from the one discussed previously, since we are now concerned with nodes that are along multiple paths between the different source nodes. The algorithm is shown as Algorithm 4. Each of the source nodes construct their neighborhoods as in the single source case. Subsequently, the nodes at the intersection that have some level of involvement in at least k of the neighborhoods are identified. Activation is performed from these nodes, with amounts proportional to their importance in the different neighborhoods. We show an example of a multi-source VPN from the DBLP graph in Fig 6. We use three source nodes, shown in yellow - Xifeng Yan, Adam Silberstein and Jessica Lin. The red nodes indicate the members of the multi-source VPN and they are labeled with their commit-

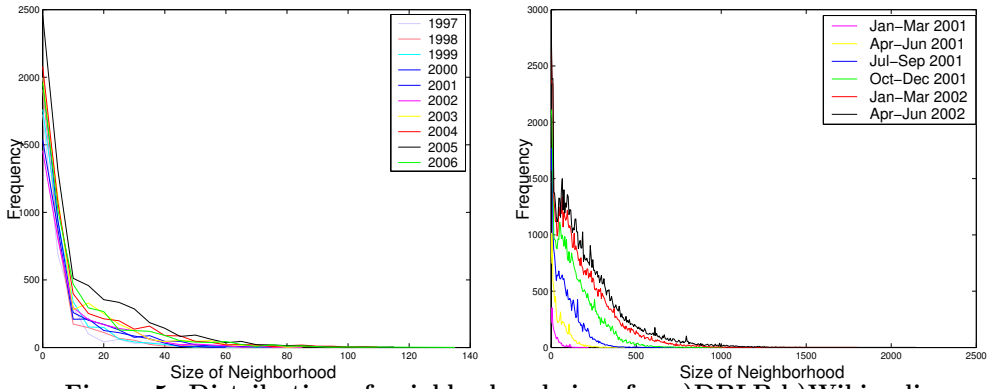


Figure 5: Distribution of neighborhood sizes for a)DBLP b)Wikipedia

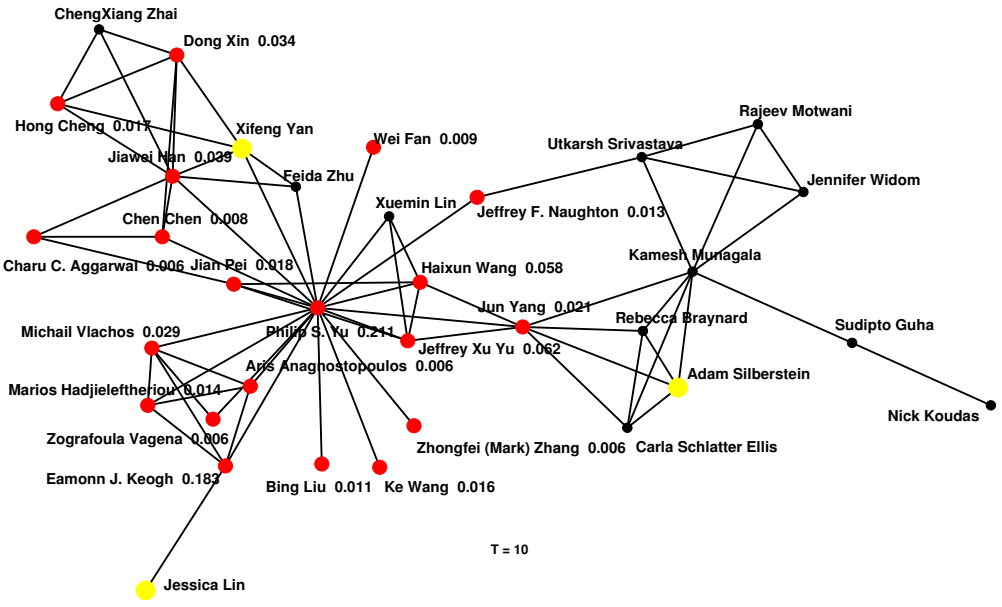


Figure 6: Example of a multi-source neighborhood. Members of the multi-VPN are shown in red.

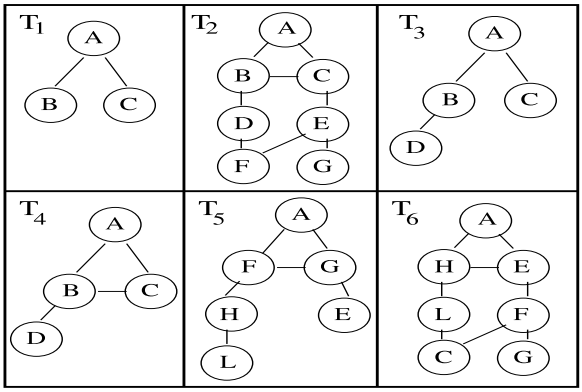


Figure 7: Illustration of events in a VPN rooted at A.

ment or importance values. We see that Philip Yu is the most central node in terms of these source nodes, and it is activated with higher values than the others. Note that the values represent importance in the shared neighborhood and not the neighborhood they are most involved in. Other nodes that are shown are nodes that exist in the individual neighborhoods but do not have a reasonable commitment value (> 0.005) with respect to the multi-VPN.

5. EVOLUTION OF VPNS

An interaction graph G is said to be *evolving* if its interactions vary over time. Let $G = (V, E)$ denote a temporally varying interaction graph where V represents the total unique entities and E the total interactions that exist among the entities. We define a temporal snapshot $S_i = (V_i, E_i)$ of G 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, new nodes and edges can appear. Similarly, nodes and edges can also cease to exist. This dynamic behavior of a graph over time can thus be represented as a set of S equal, non-overlapping temporal snapshots.

We are interested in understanding how neighborhoods evolve over time, in particular identifying key changes that occur and discovering motivations for these changes. For each snapshot S_i, there can be a set of $|N_i|$ Viewpoint neighborhoods represented as $N_i = \{N_i^1, N_i^2, \ldots, N_i^{k_i}\}$. To characterize the changes occurring in neighborhoods over time, we require the use of certain measures, which we term critical events. These events capture the behavior of nodes and neighborhoods over time. In an earlier work [3], we composed events for clusters. Here, we consider Viewpoint neighborhoods and make use of the importance and depth information to quantify key changes. Note that, the correspondence across snapshots is not an issue here, since we are looking at viewpoints of certain nodes over time, as opposed to clusters. Hence, we consider successive viewpoints for a node and find events across pairs of them.

5.1 Events for Viewpoint Neighborhoods

We define six basic events for neighborhoods. We use the VPN shown in Fig 7 to illustrate the events we describe.

- **Growth** : This event captures the size of the neighborhood increasing over time.

  $Growth(N_i^k) = 1$ iff $|V_i^k| < |V_{i+1}^k|$

  Growth in a VPN indicates that more nodes are invested in the viewpoint of a particular node. In Fig 7 at T_2, the size of the neighborhood of A increases.

- **Shrinkage** : This is the opposite of the above event and signifies the reduction of the size of a node's neighborhood.

  $Shrinkage(N_i^k) = 1$ iff $|V_i^k| > |V_{i+1}^k|$

  Shrinkage can be caused by either edge deletions in the node's immediate neighborhood or the deletion of an influential hub node in its VPN. In Fig 7 at T_3, the size of the neighborhood of A decreases.

- **Continuity** : A VPN is said to continue if the members of the neighborhood do not change. Note that, this does not place any restrictions on the link structure among the members. It conveys the information that the nodes invested in this particular neighborhood remain unchanged.

  $Continuity(N_i^k) = 1$ iff $V_i^k = V_{i+1}^k$

  Since the evolution of an interaction graph does not uniformly affect all nodes, this event serves the purpose of determining the range of such changes within the graph. If a node's neighborhood satisfies a $Continuity$ event, it demonstrates that the changes occurring in the graph do not affect this particular node in any way. Note that this is a stability measure for a node. In Fig 7 at T_4, the nodes in the neighborhood of A remain the same as the previous timestamp. Note, that there is an edge now between nodes B and C.

- **Mutate** : This event is the opposite of the above event and indicates major changes within the Viewpoint neighborhood of a node. If more than half of the members of a node's VPN are different over two successive snapshots, it indicates significant change in the VPN and hence can be considered a $Mutate$ event.

  $Mutate(N_i^k) = 1$ iff $|V_i^k \cap V_{i+1}^k| < 0.5 * |V_i^k|$

  Using this event, one can identify nodes whose neighborhoods are affected severely by changes occurring in the graph over time. Note that, a node's sociability can be quantified based on the number of $Mutate$ events, the node participates in. We provide more details at the end of this section. In Fig 7, at T_5, we find drastic changes in the VPN of A indicating a $Mutate$ event.

- **κ-Attraction** : This event signifies positive change in the Viewpoint neighborhood of a node with $\kappa\%$ of the nodes moving closer than before. Let $Dep(m)_i^k$ represent the depth (minimum distance from the root) of node m in the VPN of k at time i.

  $$S = \{m \in V_i^k | Dep(m)_i^k > Dep(m)_{i+1}^k\}$$

  $$Att(N_i^k, \kappa) = |S| iff (|S| > \kappa * |V_i^k|)$$

  If a node experiences this event, it reflects positively on the influence of that particular node. In the example, at T_6, we find that nodes H, E and L are closer to node A than previously. This signifies an $Attraction$ event on the part of A.

- **κ-Repulsion** : This event is the opposite of the previous event. It signifies the increase in distance between

a node and the members of its VPN in the previous time stamp. If $\kappa\%$ of the nodes in the VPN of a node x are farther away in the next timestamp, x is considered to partake in this event.

$$S = \{m \in V_i^k | Dep(m)_i^k < Dep(m)_{i+1}^k\}$$

$$Rep(N_i^k, \kappa) = |S| iff (|S| > \kappa * |V_i^k|)$$

This event demonstrates a negative influence of the node in question. It intrinsically represents the fact that the changes, that are occurring in the graph as a whole, have an adverse effect on the relations of this node with its neighbors. In the figure, at T_6, the nodes F and G which were close to A are now at a greater distance from A, indicating a *Repulsion* event.

Note that the events we describe above are not mutually exclusive. For instance, it is common for a neighborhood to undergo a *Growth* event and an *Attraction* event at the same time. To find the events, we consider two snapshots of VPNs at a time. We build an index on the root of the neighborhoods, so that correspondence is not an issue. We compare the corresponding neighborhoods of a node to identify all events for that node.

The number of events discovered for the two datasets are shown in Table 1. We can observe that in DBLP, the size-based events (Growth and Shrinkage) are both frequent, while Continue events are rare. In the case of Wikipedia, Growth and Attract events far outnumber Shrinkage and Repel events. This is due to the fact that in Wikipedia, nodes (pages) and links are added but not frequently deleted, which causes neighborhoods to increase rather than decrease. Also, Continue events are quite frequent in Wikipedia, suggesting that semantic neighborhoods do not change much over time.

5.2 Behavioral Measures

We can use the events described in the previous subsection to build behavioral measures to signify key behavioral patterns that occur over time. We define three measures - Stability, Sociability and Popularity as follows. Note that, it is possible to define measures for capturing other types of behavior as well using the above mentioned events.

5.2.1 Stability

Stability is a measure of how the changes to the graph affect a particular node. If a node's principal neighborhood (VPN) does not change much over time, then it is believed to be stable.

$$Stability(x) = \frac{\sum_{i=1}^{T} Continuity(N_i^x)}{|Activity(x)|} \quad (3)$$

Here, $Activity(x)$ denotes the number of pairs of successive timestamps this particular node is active in. We used the measure shown above on authors of the DBLP dataset. We found the authors with top values of this measure. The author who had the highest stability score was Juho Rousu and the second author was Tapio Elomaa. When we examined the DBLP bibliography entry for Prof Juho Rousu, we found that from 1996 to 2003, every paper that this author published [3] was with Prof Tapio Elomaa. Hence, the fact that these two authors are at the top of the Stability list is justified.

5.2.2 Sociability

Sociability is a measure of how many different nodes are affected by or cause effect to this particular node over time. It can be described using the $Mutate$ event as follows:

$$Sociability(x) = \frac{\sum_{i=1}^{T} Mutate(N_i^x)}{|Activity(x)|} \quad (4)$$

It is calculated as the ratio of the number of timestamps its neighborhood changes drastically to the number of pairs of successive timestamps it is active in.

5.2.3 Popularity

Popularity is a measure of how many nodes are attracted to the node's neighborhood. It can be described using the κ-Attraction and κ-Repulsion events.

$$Popularity(x) = \frac{\sum_{i=1}^{T} Att(N_i^x, \kappa) - Rep(N_i^x, \kappa)}{|Activity(x)|} \quad (5)$$

If a node attracts several nodes over time and does not have high repulsion rates, it is considered popular. In the case of Wikipedia, popularity reflects a buzz around a particular topic page, as more pages and links are added to it. This buzz can be identified by a spike in the popularity trend graph. We computed the popularity for VPNs (as $Att(N_i^x, \kappa) - Rep(N_i^x, \kappa)$) at different timepoints. We identified interesting real-world events in the 2001-2002 period and analyzed the corresponding trend plots. Note that, we are not considering new pages created based on new events. An example for this would be the September 11 attacks (which did have high popularity when created). We consider neighborhoods that already existed but spiked at a particular timepoint, indicating a buzz.

The trend plots are shown in Table 2. The event that inspires the popularity is shown in the first column. The root nodes of the VPNs under consideration are presented in Column 2 with their corresponding popularity scores in the subsequent columns. The value spikes at the time corresponding to the particular event in each case.

Note that, the above measures can be computed incrementally from the events discovered at each timestamp.

5.3 Impact on Node Neighborhoods

We would like to consider which nodes have high impact on most VPNs. This would also allow us to verify our activation model of assigning importance or committment values to nodes with respect to neighborhoods. Our hypothesis is that the nodes that impact or influence a neighborhood would have high importance values for that particular neighborhood. Since, a node may be involved in different neighborhoods in differing capacities, we define the impact or influence of a node as the weighted sum of its committment values in all neighborhoods it is a part of.

$$Impact(x) = \sum_{i=1}^{T} \sum_{k=1}^{|N_i|} P(x, N_i^k) \quad (6)$$

[3] in the conferences we considered

	DBLP			Wikipedia		
Time	Growth/Shrinkage	Continue/Mutate	Attract/Repel	Growth/Shrinkage	Continue/Mutate	Attract/Repel
1-2	434/347	58/743	426/377	1146/16	570/867	851/12
2-3	527/428	75/855	473/403	6640/256	1409/4799	4543/171
3-4	500/404	49/893	491/484	19773/869	2628/16563	15783/877
4-5	540/437	60/914	525/502	39410/3646	3273/27051	22487/2319
5-6	450/466	40/849	474/549	51899/7718	9561/19135	13532/1579
6-7	587/474	42/1059	636/662	65683/10695	7189/34880	26487/4155
7-8	739/664	44/1396	858/851			
8-9	947/681	57/1617	1037/976			
9-10	672/930	37/1556	762/1088			

Table 1: Event Occurrences for DBLP and Wikipedia. For Attrack and Repel we use κ of 0.5.

Event	VPN Root	Apr-Jun 2001	Jul-Sep 2001	Oct-Dec 2001	Jan-Mar 2002	Apr-Jun 2002
Jun 1 - Nepal Royal Family Massacre	Nepal	0	111	0	0	0
Jun 20 - Pervez Musharraf becomes president of Pakistan	Pervez Musharraf	0	100	0	0	0
	Politics of Pakistan	0	35	0	0	0
	History of Pakistan	0	68	0	0	0
Aug 25 - R&B singer Aaliyah dies in a plane crash	Aaliyah	0	281	0	298	0
Sept 11	Terrorism	0	107	68	0	0
	Patterns of Global Terrorism	0	106	98	0	140
	Osama Bin Laden	0	0	92	116	35
	World Trade Center	0	0	53	0	0
	Islamist Terrorism	229	268	137	0	0
Sept 18-Oct 9 Anthrax attacks using letters	Anthrax	0	0	248	0	0
Dec 19 - Lord of the Rings: Fellowship of the ring released in US	Fellowship of the Ring	0	0	27	31	0
	Peter Jackson	0	0	226	0	506
	J.R.R Tolkein	0	95	105	0	0
Dec 22 - Hamid Karzai sworn in as President of Afghanistan	Democratic Republic of Afghanistan	0	0	70	257	0
	Foreign Relations of Afghanistan	0	0	316	0	0
Mar 24 - Oscars	74th Academy Awards	0	0	0	280	95
	A Beautiful Mind	0	0	0	96	0
	Denzel Washington	0	0	400	850	744
	Halle Berry	0	0	0	177	681
	Jennifer Connelly	0	0	0	1132	79
June - Serena Williams wins Wimbledon	Wimbledon	0	0	0	214	275
	Serena Williams	0	0	0	0	50

Table 2: Popularity Trends in Wikipedia.

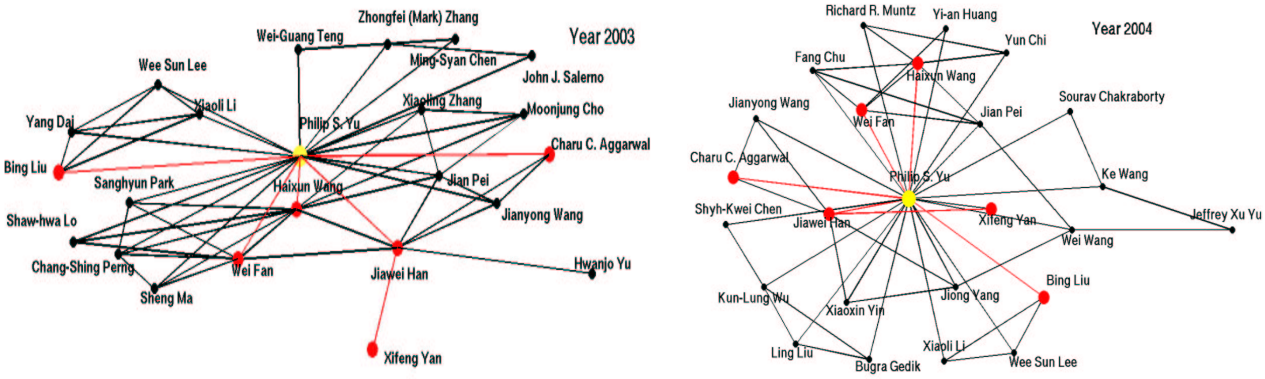
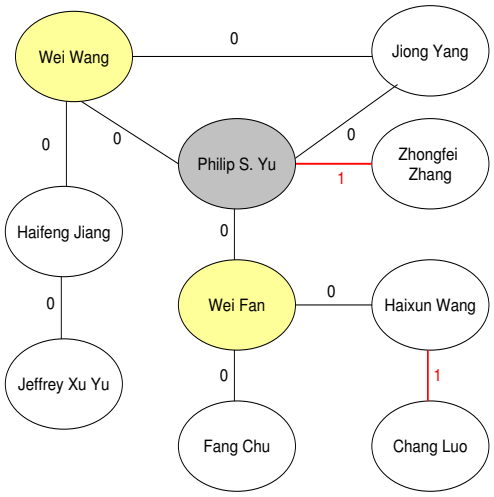


Figure 9: Two snapshots (2003 and 2004) of Philip Yu's evolving VPN. The core subgraphs are shown in red.



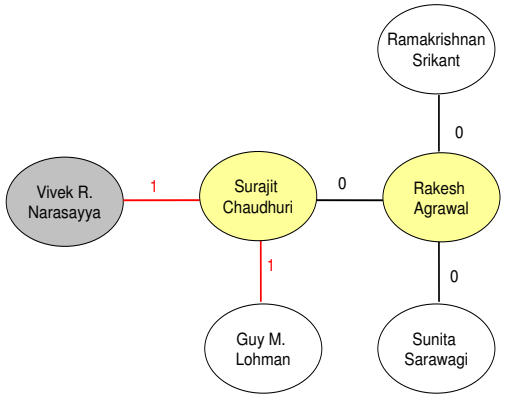


Figure 10: Largest maximal frequent transformation subgraphs in the VPNs of a) Philip S. Yu b) Vivek R. Narasayya.

Impact
Philip S. Yu
Jiawei Han
Elke A. Rundensteiner
Christos Faloutsos
Hans-Peter Kriegel
Surajit Chaudhuri
Divesh Srivastava
Daphne Koller
Raghu Ramakrishnan
Beng Chin Ooi
Divyakant Agrawal
Sebastian Thrun

Table 3: Top 12 Values for the Impact measure

When we computed this quantity for the DBLP co-authorship dataset, we observed that most of the authors who had high values were authors who could be considered influential in terms of their research in the community. A list of the authors having the top 12 values for this quantity is given in Table 3. This provides some verification for the values that we assign in the activation model.

5.4 Core Subgraphs

Note that an advantage of using VP-neighborhoods is that, it enables us to perform frequent pattern mining over the viewpoint neighborhoods of points. By examining the VP-neighborhoods for a point over all time instances, we can identify core substructures using frequent graph mining techniques.

Definition: We define a *core subgraph* for a given source node as the largest subgraph in its Viewpoint Neighborhood that is frequent over time.

In the context of collaboration networks, a frequent subgraph or subtree in the VP-neighborhoods of a graph indicates a core group associated with a particular author. By finding these core substructures, we can also gauge the level of stability for an author in terms of their neighborhood. An absence of significant core subgraphs would indicate disparate behavior with different groups separated

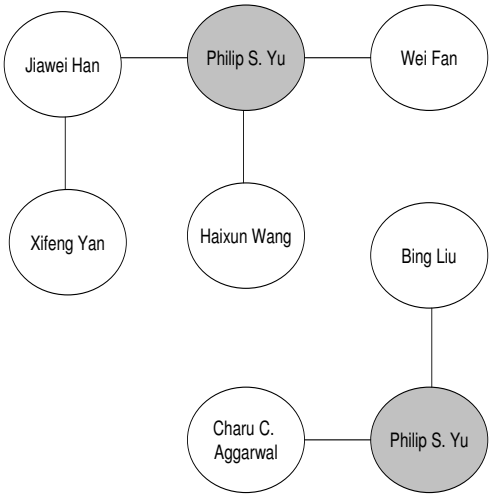


Figure 8: Maximal core subgraphs in the VPNs of Philip S. Yu.

Jeffrey Xu Yu and Hongjun Lu
Wei Wang and Jiong Yang
Philip S. Yu and Jiong Yang
Philip S. Yu and Wei Wang
Philip S. Yu and Wei Wang — Philip S. Yu and Jiong Yang
Surajit Chaudhuri and Vivek R. Narasayya
Jiawei Han and Jian Pei
H. V. Jagadish and Raymond T. Ng
H. V. Jagadish and Laks V. S. Lakshmanan
H. V. Jagadish and Laks V. S. Lakshmanan
Divesh Srivastava and Laks V. S. Lakshmanan
Divesh Srivastava and H. V. Jagadish
Divesh Srivastava and Nick Koudas

Table 4: Frequent transformation subgraphs for the DBLP dataset with only insertions shown. High-support edges are shown in bold.

from each other. We would like to identify and examine these core subgraphs in VP-neighborhoods. To illustrate, we selected Dr Philip S. Yu, who is an influential and popular author. We computed the core subgraphs [4] for the VPNs of Dr Yu. We used a support threshold of 5 to obtain 2 maximal frequent core subgraphs. The corresponding core subgraphs are shown in Fig 8. We also illustrate Dr Yu's VPN over two timestamps in Fig 9, showing the core subgraphs in red. When we chose Juho Rousu, the author with the highest stability measure value, we got a frequent subgraph *Juho Rousu and Tapio Elomaa* with very high support (7/9).

5.5 Transformation Subgraphs

Apart from characterizing changes occurring in the graph over time, our goal is to also reason about the effect of changes on nodes and neighborhoods. As we mentioned earlier, changes in the graph are likely to affect different nodes in different ways. We are interested in identifying influential transformations that affect most of the nodes in the graph. For this, we again leverage the use of frequent subgraph mining over VPNs. The influential transformations, that we are interested in, are those that affect a majority of VPNs. By mining the changes occurring in VPNs over time, we can identify such transformations.

First, we need to represent the changes occurring in a particular VPN over a set of snapshots. For this we introduce a $Transformation\ Subgraph$. A transformation subgraph TS for a VPN over two timepoints i and $i+1$, is a graph consisting of only edges that belong to the neighborhood either at time i or time $i+1$ but not both. $TS(N_i^k, N_{i+1}^k) = (V_{TS}(N_i^k, N_{i+1}^k), E_{TS}(N_i^k, N_{i+1}^k))$ where $E_{TS}(N_i^k, N_{i+1}^k) = (E_i^k \oplus E_{i+1}^k)$

However, we also need to distinguish edges that were inserted during the timestamp from edges deleted. For this purpose, we use edge labels, labeling each edge in the transformation subgraph as 0 (deleted) or 1 (inserted).

We can represent the evolution of a VPN as a time-series of transformation subgraphs, each representing changes over a pair of snapshots. Subsequently, we can perform frequent subgraph mining to identify the key transformations that affect the viewpoint neighborhoods of most of the nodes. We performed frequent subgraph mining on the DBLP graph, using a support threshold of 25% which resulted in 23 unique transformation edges [5]. We list the frequent subgraphs that affected the most VPNs when inserted, in Table 4. Out of the 13 shown, 2 edges *Divesh Srivastava and H. V. Jagadish* and *Jiawei Han and Jian Pei* were frequent with the highest support (causing change in $>$ 150 VPNs) over the 10 timestamps. Also, one can identify frequent transformation subgraphs for individual nodes. These represent subgraphs or edges that have a high effect on the neighborhood of a particular node over time. We consider the VPNs of two authors Philip S. Yu and Vivek R. Narasayya. The largest maximal frequent transformation subgraphs in their VPNs are shown in Fig 10. We can see that the authors with high-degree are typically influential authors (shown in yellow), which explains the effects they have on the source neighborhood. An important difference that can be observed is that Philip S. Yu has high degree in his own transformation subgraph, which indicates that his own interactions cause most of the changes in his neighborhood. On the contrary, Surajit Chaudhuri and Rakesh Agrawal play an important role in affecting the VPN of Vivek Narasayya.

[4]We use the Graph mining toolkit developed by Gregory Buehrer for this purpose [8]

[5]some transformation edges were frequently added as well as deleted

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed an activation model to construct the Viewpoint Neighborhood of a node and quantify the relationships that exist within it. We have also shown how a common neighborhood of interest can be computed for a group of nodes and how different activation functions leveraging topological and semantic information can be constructed to facilitate the extraction of interesting neighborhoods. To characterize and measure the effect of changes over time, we have introduced temporal events as well as behavioral measures that can be computed incrementally. Finally, the use of frequent subgraph mining to identify stable and fleeting subgraphs has been highlighted. The algorithms and analysis provided are particularly relevant for social network applications such as personalized and community search, and online advertising.

In future work, we would like to extend the temporal analysis to graph production rules and graph grammars. Of particular interest in our context will be to evaluate if graph grammars can be inferred from such interaction networks by learning the *production rules* that govern evolution of clusters or viewpoint neighborhoods.

7. ACKNOWLEDGEMENTS

This work is supported in part by the NSF CAREER Grant IIS-0347662 and NSF SGER Grant IIS-0742999.

8. REFERENCES

[1] F. Alkemade and C. Castaldi. Strategies for the diffusion of innovations on social networks. *Computational Economics*, 25(1-2), 2005.

[2] S. Amer-Yahia, M. Benedikt, and P. Bohannon. Challenges in searching online communities. *IEEE Data Eng. Bull.*, 30(2):23–31, 2007.

[3] S. Asur, S. Parthasarathy, and D. Ucar. An event-based framework for characterizing the evolutionary behavior of interaction graphs. *SIGKDD*, pages 913–921, 2007.

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

[5] A. L. Barabasi, H. Jeong, Z. Neda, E. Ravasz, A. Schubert, and T. Vicsek. Evolution of the social network of scientific collaborations. *Physica A: Statistical Mechanics and its Applications*, 311(3-4):590–614, August 2002.

[6] K. Berberich, S. Bedathur, T. Neumann, and G. Weikum. A time machine for text search. In *SIGIR*, New York, NY, USA, 2007. ACM.

[7] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[8] G. Buehrer, S. Parthasarathy, A. Nguyen, D. Kim, Y. Chen, and P. Dubey. Towards data mining on emerging architectures. *SIAM Workshop on High*

Performance and Distributed Mining (HPDM06), 2006.

[9] R. Cowan and N. Jonard. Network structure and the diffusion of knowledge. *Journal of Economic Dynamics and Control*, 28:1557–1575, 2004.

[10] D. Crandall, D. Cosley, D. Huttenlocher, J. Kleinberg, and S. Suri. Feedback effects between similarity and social influence in online communities. *SIGKDD*, 2008.

[11] C. Faloutsos, K. S. McCurley, and A. Tomkins. Fast discovery of connection subgraphs. In *SIGKDD*, 2004.

[12] C. L. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977.

[13] E. Gabrilovich and S. Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. *International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

[14] V. Kacholia, S. Pandit, S. Chakrabarti, S. Sudarshan, R. Desai, and H. Karambelkar. Bidirectional expansion for keyword search on graph databases. *VLDB*, pages 505–516, 2005.

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

[16] D. Kempe, J. Kleinberg, and E. Tardos. Influential nodes in a diffusion model for social networks. *Proc. Intl. Colloquium on Automata, Languages and Programming (ICALP)*, 2005.

[17] R. Kumar, J. Novak, and A. Tomkins. Structure and evolution of online social networks. In *SIGKDD*, 2006.

[18] J. Leskovec, L. Backstrom, R. Kumar, and A. Tomkins. Microscopic evolution of social networks. In *SIGKDD*, 2008.

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

[20] G. Li, B. C. Ooi, J. Feng, J. Wang, and L. Zhou. Ease: an effective 3-in-1 keyword search method for unstructured, semi-structured and structured data. *SIGMOD*, 2008.

[21] 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.

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

[23] T. Snijders. Models for longitudinal network data. *Book chapter in Models and methods in social network analysis, New York: Cambridge University Press*, 2004.

[24] J. Sun, C. Faloutsos, S. Papadimitriou, and P. S. Yu. Graphscope: parameter-free mining of large time-evolving graphs. In *SIGKDD*, pages 687–696, 2007.

[25] C. Tantipathananandh, T. Y. Berger-Wolf, and D. Kempe. A framework for community identification in dynamic social networks. *SIGKDD*, pages 717–726, 2007.

[26] H. Tong and C. Faloutsos. Center-piece subgraphs: problem definition and fast solutions. *SIGKDD*, pages 404–413, 2006.

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