Year 4 Final Project Report 2015-2016

A National Science Foundation
University Cooperative Research Center

Graph Sampling, Summarization, and Touch-Based Visual Analytics for Large Complex Systems

Contents

Personnel

Principal Investigator:

Christoph Borst PhD and Mehmet Engin Tozal PhD, University of Louisiana at Lafayette

Graduate Students:

Nicholas Lipari and Maryam Heidari, PhD students at University of Louisiana at Lafayette

Executive Summary/Abstract

Objectives

Our project enables the interactive visual analytics of large-scale graphs using novel graph sampling methods and touch-based interfaces. Recently, there is a significant interest in modeling and studying real-world complex systems as large-scale graphs with numerous interconnected or interacting entities. Many real-world systems such as online social networks (OSN), world wide web (WWW) and Internet topology maps (ITMs) are very large, so capturing them in their entirety, analyzing them to extract useful information, and visualizing them for decision making are resource-consuming and challenging tasks [1]. It is necessary to develop graph sampling approaches and integrate them with human-computer interfaces to study these large-scale graphs to understand the underlying real-world systems. These networks are large and decentralized which make the global structure of them invisible [2]. One of the approaches to overcome this issue is sampling. In sampling small subsets of nodes and links from a network are selected. Sampling, as illustrated in Figure 1, makes it possible to study a small part of the networks while preserving features of the original network. There are many related works to graph sampling, however, they target different types of networks context and they have different characteristics [3] [4]. It is necessary to develop graph sampling approaches and integrate them with human-computer interfaces to study these large-scale graphs to understand the underlying real-world systems.

Methods

The PIs investigated sources of information loss in a graph sampling process and identified fundamental factors that need to be carefully considered in a sampling design. We also developed a software system as an extension to open source libraries (igraph [5] and Boost [6]) that employs different sampling methods to estimate important graph characteristics. Developing an extension to these libraries, instead of a standalone application, allows more seamless integration of our work with other CVDI projects. Furthermore, the project improved interactive visual analysis of large graphs by prototyping interface methods in combination with machine analytics. We developed multitouch and gesture techniques to provide intuitive user

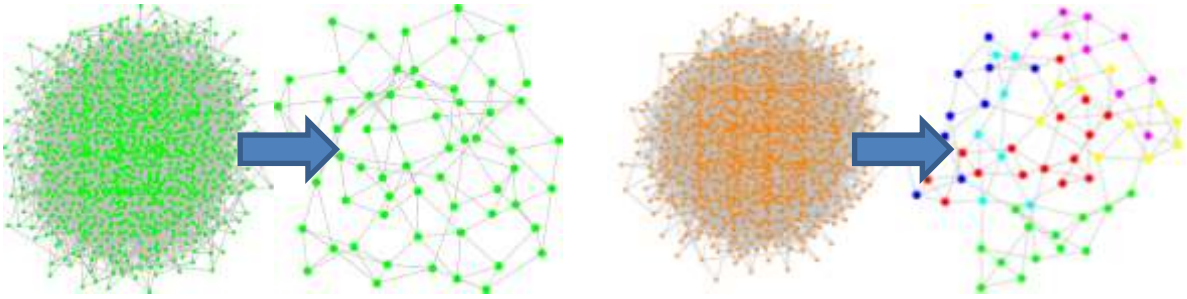

Figure 1: Conceptual examples of graph sampling. *(Left)* From a large-scale and dense graph, we can extract structural and functional characteristics of underlying systems. *(Right)* A network of customer relationships can be sampled to highlight influential customers, and the interactions among these entities appear more readily. In this example, customer communities are identified by their common color.

control of navigation, filtering, clustering, and highlighting during visual analysis. The efficiency and clarity of interfaces is critical for the success of visual analytics systems and helps users understand results and analysis processes.

Each use-case is different, but we have evaluated two examples of real-world data in developing such an approach. In a HEP-TH (high energy physics theory) citation graph [7], nodes represent research papers and edges are present when one paper references another. The dataset contains 27,770 papers with 352,807 total citations, all of which were published between January 1993 and April 2003. The other was a human disease co-occurrence dataset [8] where nodes consisted of 16,459 diseases labeled by their ICD-9 codes. Edges represented 6,088,553 co-occurrences of diseases diagnosed between 1990 and 1993.

Results

We have studied graph sampling and visualization for large network datasets. Two newly-developed sampling approaches achieve biased and unbiased sampling, respectively. We tested these against standard selection and walk-based sampling. The sampling module has been integrated into the analytics server portion of our visual analytics framework. The analytics server has also been extended to include several CDF plotting options and more rigorous thread management. The visualization station allows users to control sampling parameters, view sampled datasets, create summary plots of graph centrality metrics, and save sampled graphs upon completion.

Conclusions

We estimated how some sampling techniques maintain properties of the original graphs meanwhile others result in biased sample graphs. The unbiased samples are more suitable for experiments or data mining, while biased sampling can draw attention to certain nodes (e.g.,

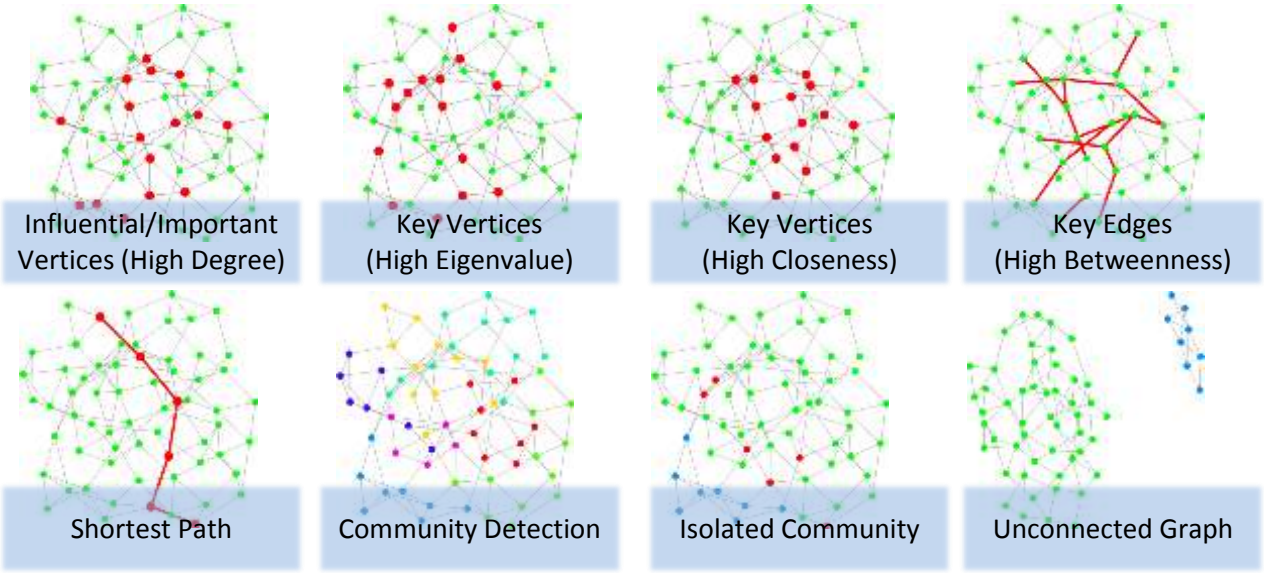

Figure 2: Examples of Sampled and Summarized Graph Analysis.

high degree) and is generally better suited for visualization. The visualization station includes functionality to improve the user's role in the analytics process. An interactive sampling mechanism allows users to choose sampling type and parameter (e.g., percentage of edges from a graph). Such a tandem exercise allows users to more quickly arrive at their intended view of the data through objective and subjective criteria. The examples in Figure 2 show some criteria currently used and of future interest for our research.

Differences from Current State of Art

Examples of Graph Sampling and Visual Analytics

The field of visual analytics has produced several works aimed at graph analysis. GraphViz [9] and Network Repository [10] count among those combining graph sampling with visual analytics. A web-based graph analytics system uses edge sampling with Sample and Hold strategy to support streaming graphs. Their algorithm maintains a small amount of state and samples over the streaming graph. A survey [11] found multiple methods of selection and interaction, as well as multiple analogies for manipulating virtual objects. The authors suggest a framework centering on interaction feedback, physically-based virtual object movements, and an understanding of prior experience of users. TouchWave [12] and TouchViz [13] allow users to manually interact with time-series data in chart form. TouchWave attempts to improve the legibility of chart data, the ability to make comparisons, and the scalability to interpret dense datasets.

Graph Sampling is a technique to pick a subset of nodes and edges from original graph. Sampling has a wide range of applications [4], such as surveying hidden populations in

sociology, graph visualization, scale down Internet Autonomous System (AS) graphs, graph sparsification. Some modern examples of graph sampling are based on graph topology. Breadth-First Sampling (BFS) [14] explores the neighbor nodes first, before moving to the next level neighbors, while Depth-First sampling (DFS) [15] selects the latest explored but not-visited node. Forest Fire (FF) which is a randomized version of BFS, selects the next neighbor node uniformly, while Snowball Sampling chooses several neighbors randomly out of all neighbors of a node [16]. The most common approach of this family is Random Walk (RW) [17], which chooses nodes from a neighborhood based on a uniformly random process. A verified version of random walk is the Metropolis Hastings algorithm [18], in which the next node is chosen uniformly at random among the neighbors of the current node.

Uniqueness

There are some challenges in studying complex networks. Capturing them in their entirety, analyzing them to extract useful information, and visualizing them for decision making are resource-consuming and challenging tasks. To assist in understanding the underlying mechanisms, we have developed a graph analytics system. The uniqueness of this approach lies in the combination of graph sampling with interactive visualization, wherein users can interactively explore their datasets while requesting different samples and analysis. Our system is illustrated in Figure 3 and is separated into two components: an analytics server and a visualization station. The two pieces share an inter-process communication protocol. The analytics server is responsible for reading graphs from a data store, sampling graphs, and computing topology metrics. The visualization station renders graph objects and analytics results, receives touch input from users, and communicates those inputs as analytics requests.

Offline, unbiased sampling is useful in various domains. The global structure of large networks is not fully visible. Unbiased sampling enables us to sample proprietary features of population networks and generalize the results. Analysis can then proceed without providing any further access to the population network. We have proposed a new unbiased sampling algorithm, which is based on path sampling. The unbiased, re-weighted path sampling algorithm is effective to infer structural properties of the graph itself. The unbiased sampling provides scalability and extensibility. With a sample graph, it is now computationally affordable for various graph algorithms to compute essential properties of the original graph (e.g., degree distribution, betweenness centrality and clustering coefficient). These properties can also be plotted along with the node-link diagram. However, biases of certain sampling methods can be useful: extracting features to characterize nodes and investigate the local neighborhood of selected nodes.

Results

Methods (System Framework)

The analytics server supports such data sources as SQLite [19] and Neo4j [20]. To avoid loading the entire graph into memory, we sample from the data source and store only a portion of the graph as an igraph [5] object. The resulting graph can be tuned based on sampling parameters (e.g., percentage of nodes). From the in-memory sample graph, we extract edge and node lists, calculate graph metrics, and perform analytics processes. The graph information is sent to the visualization station over a communication protocol. We also demonstrate how choice of sampling algorithms can produce either unbiased or biased sample graphs (e.g., choosing nodes with a high degree).

The visualization station presents dataset, sampling, and visualization choices to the user. Interactions can be either touch gestures or mouse clicks. The system interprets these actions and issues requests to the analytics server. In one common sequence of actions, a user selects a dataset and sampling type. The analytics server performs the sampling algorithm, while simultaneously sending progress amounts. Upon completion, the visualization server requests edge and node data about the sample and stores these in a form efficient for rendering.

Methods (Sampling Techniques)

Analytic module includes the graph sampling approaches and their dependencies. We use four different sampling algorithms as the basic algorithms which includes node sampling, edge sampling, random walk and metropolis hasting. Moreover, we have proposed two different

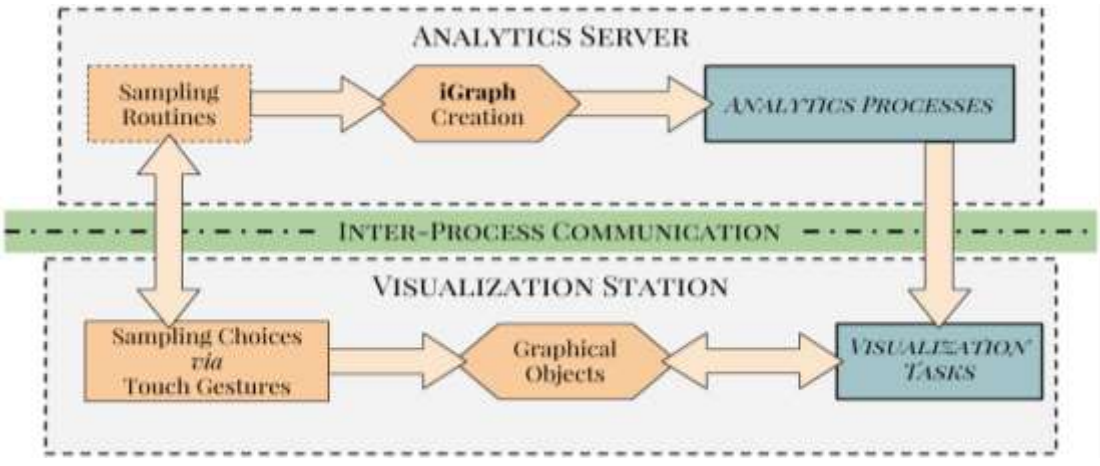

Figure 3. Flowchart of Analytics Server and Visualization Station communication. Users interact with touch gestures to select sampling choices and parameters. Sampled graphs are stored in an igraph object on the Analytics Server. Users may also perform various visualization tasks, which spawn corresponding analytics processes. The Visualization Station renders the results as graphical objects (e.g., node-link diagrams and CDF plots)

algorithms to achieve both biased and unbiased sampled graph. In this section we discuss the new algorithms.

A. Biased Sampling Algorithm

The random walk with random jump algorithm is a modified version of random walk. The proposal of this algorithm is to achieve a biased sample of the original graph. In this project the biased sampled graph is a sampled graph that has the nodes with high betweenness centrality in original graph. Betweenness centrality is an indicator of a node's centrality in a network. It is equal to the number of shortest paths from all nodes to all others that pass through that node. It is a global centrality index that quantifies how much a node controls the information flow between all pairs of nodes in a graph. In this algorithm we pick a starting node uniformly at random. With probability p, choose an outgoing edge uniformly at random and with probability (1/p), perform a jump to a random starting node and restart the random walk. Also if in the process of random walk, the algorithm chooses a stub node, it performs the random jump. Stub node is a node with no outgoing connection.

B. Unbiased Reweighted Path Sampling

In this section we discuss the proposed unbiased path sampling algorithm. The path sampling algorithm is a traversal sampling algorithm in which the sampler starts with a two random nodes and expand the sample based on current observations. In path sampling we expand the sampled graph based on finding the shortest path among the two random nodes. The algorithm continues by picking two random nodes and finding the shortest path between them. In each step, it adds the new nodes and new edges into the sampled graph.

The algorithm consists of two main parts, (1) finding shortest path p, (2) re-weighting edge of p. As we know, a path in an undirected graph is a sequence of nodes $P = (v_1, v_2, \ldots, v_n) \in V \times V \times \ldots \times V$ such that v_i is adjacent to v_j for $1 <= i < n$. Such a path P is called a path of length n-1 from v_1 to v_n.

Let $e_{i,j}$ be the edge incident to both v_i and v_j. Given a real valued weight function $f : E \to R$ and an undirected (simple) graph G, the shortest path from v to v' is the path $P = (v_1, v_2, \ldots, v_n)$ (where $v_1 = v$ and $v_n = v'$) that over all possible n minimizes the sum

$$\sum_{i=1}^{n-1} f(e_{i,i+1})$$

When each edge in the graph has unit weight or $f : E \to 1$, this is equivalent to finding the path with fewest edges.

Following this intuition, we randomly pick two nodes and try to find the shortest path between them. The sample includes all the edges picked. The algorithm samples paths between two randomly picked nodes repeatedly until we attain a sample size.

Using path sampling to achieve a sampled graph, is likely to be biased toward nodes with high degrees. Thus, a good way to sample an un-biased graph is to apply a function which helps us to sample each path with the same probability and is thus capable of estimating graph properties without any bias. The approach that we are using is a re-weighting function.

As we mentioned the first step in algorithm is to compute the shortest path between two randomly picked nodes. Next, the edges of the original graph are reweighted using the values computed by the re-weighting function. Therefore the re-weighting method, will affect the shortest path between the next randomly chosen nodes in the next iteration.

Methods (Graph Visualization and Analysis)

The visualization station, depicted in Figure 4, sends requests to and receives graph data from the analytics server. Currently, we have implemented this inter-process communication over a

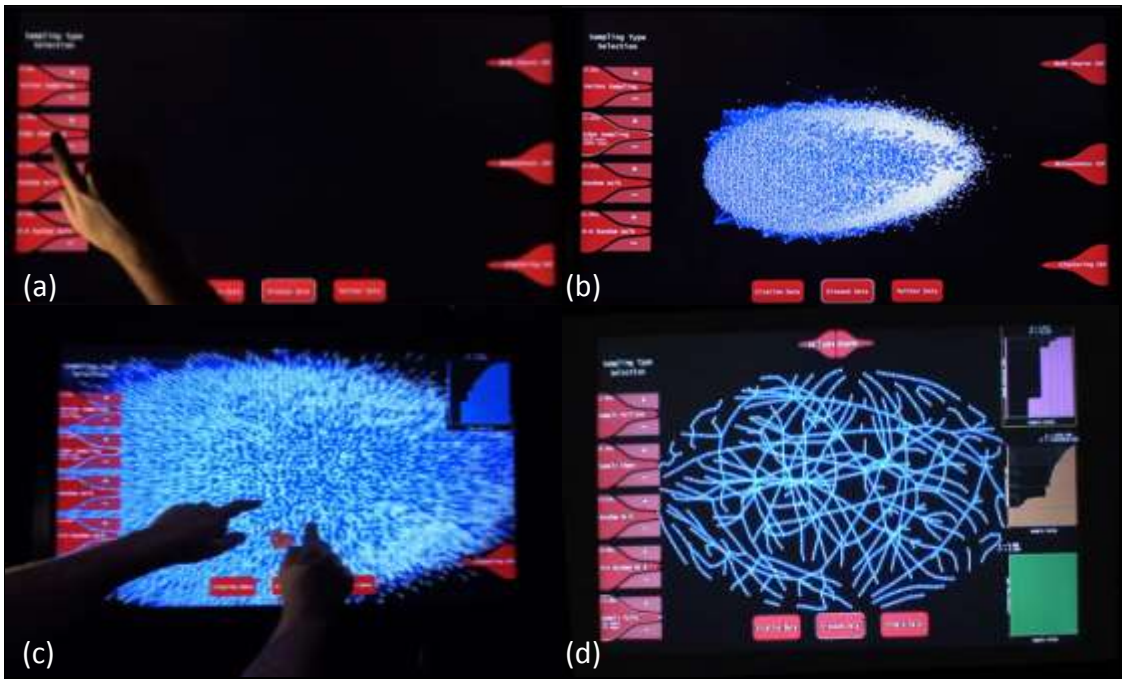

Figure 4: Creation of graph sample. *(a)* Source dataset and sampling type are selected. Sampling parameter can also be adjusted up (+) and down (-) by a constant ratio. *(b)* Graph with one percent of edges from the human disease dataset. The sample has approximately 10,000 nodes and 60,000 edges. *(c)* Two-finger navigation allows users to narrow the scope of graph to be rendered. *(d)* Path sampling with 731 nodes and 723 edges. The sampled graph is also described by the CDFs (right).

network socket. Topological information (e.g., node degree and sorted neighbor lists) and graph layouts are cached on the visualization station in a format efficient for graphical rendering. The analytics server maintains a connection to the on-disk data source and manages the in-memory samples. Graph samples, topological metrics, and layout positions are computed by separate worker threads upon request.

We have designed and implemented a multi-touch visualization station in order to provide interactive graph analysis to users. The basic navigation and selection tasks follow standard, intuitive gestures such as tap to select and pinch to zoom. Our application-specific interactions included dataset, sampling algorithm, and CDF plot creation, all accessed through a menu. Users can request more or less of the original graph by adjusting the sampling parameter. The CDF plots also act as an interaction area for users to define a range of interest for several centrality metrics (detailed below).

Once a dataset and sample have been selected, the visualization station sends a sample request to the analytics server. Progress reports (percent complete) are updated while the sampling algorithm executes. The visualization station extrapolates completion time based on time passed thus far and provides both percentage and time estimate to the user.

When the sampling algorithm completes, the visualization station automatically requests the node and edge lists from the analytics server. Two graph rendering methods are supported: pre-rendered and direct-rendered. The pre-rendered method creates a high resolution texture of the graph drawing the allow faster draw and interaction rates. We currently choose rendering method based on node and edge counts, but a more robust, hardware independent choice might consider average frame rate during idle rendering.

A. Multi-touch Gestures

Aside from the standard navigation and selecting interactions, we have tested several multi-touch interactions to improve the exploration of graphs. Graph sampling controls allow users to interact with the analytics server and achieve the most appropriate sample for their needs. To achieve a sample graph with high betweenness nodes (which are hops along many paths), users can select a path sampling or walk-based sampling algorithm. These types of samples are biased toward nodes with high betweenness.

Graph summarization is controlled by a density parameter that the user sets via a swipe interaction. Touching any node displays the label and any previously computed metric values (Figure 5, left). Users can select two points on a CDF plot to specify a range of centrality measure values (Figure 5, right). The graph highlights nodes falling within this range. The subgraph induced by these vertices can also be rendered with a menu selection.

B. Graph Summarization

As mentioned above, the sampled graph can be further simplified by a summarization routine. We have included an edge-collapsing algorithm controlled by a density parameter and scaling interactions. The graph nodes are compared against some threshold metric as users adjust these controls. To maintain interactive rates two heaps are used to determine which nodes are collapsed or expanded, depending on the change in density parameter. Each node maintains a list of incident edges from the sampled graph, a pointer to the super-node chosen during collapsing, and a list of edges collapsed onto it during collapsing.

The summarization (coarsening) by edge collapse animates nodes and edges toward their destination. We define a summarization threshold as the density parameter multiplied by the interaction scale. When the threshold increases above a node's weight (e.g. degree as used in k-core decomposition), the node collapses onto a super-node chosen as the nearest neighbor with the largest weight. Weights are recalculated and heaps are reorganized after each summarization.

Un-coarsening or expansion follows by reversing the above process based on a decreasing summarization threshold. Nodes at the top of the collapsed heap are animated away from their super-nodes, the original edges are updates, and the collapsed edges are removed from the super-node's list.

Methods (Analytics Server Integration)

A. Dataset Sources

There are different options to store and process large input/output datasets in a system. One of the common solutions to work with big datasets is to use database approaches. They can be

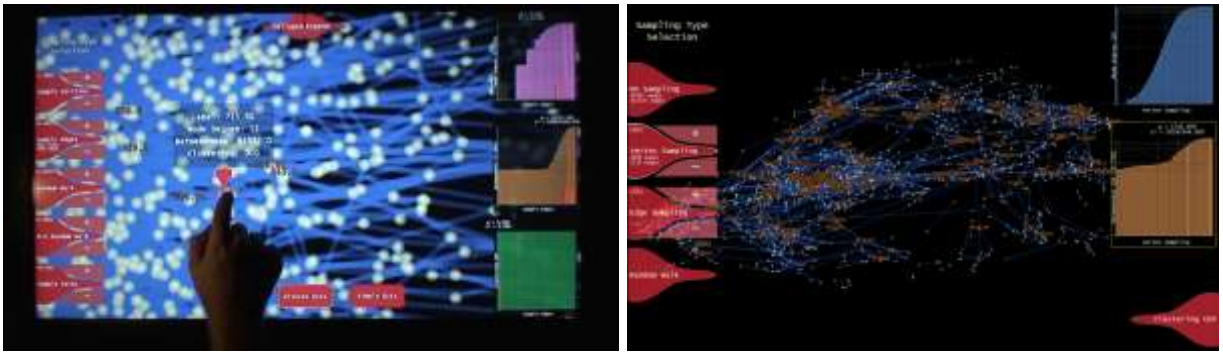

Figure 5: Graph interaction examples. *(Left)* After computing CDF plots, a user can select a node to retrieve the metrics specific to that object. (*Right)* The entire sample can be searched for nodes within a range of metric values, here showing nodes in the top 6.75% of betweenness scores. Users specify metric range with a pinch gesture.

NodeTable

nodeid	name	x	y	nlist	elist
bigint	string	float	float	string	string

nlist: list of adj nodes
n1,n2,n3,n4,....
elist: list of edges
e1,e2,e3,e3,...

EdgeTable

edgeid	weight	Nodeinfo1	Nodeinfo2
bigint	string	String	string

Nodeinfo1 : headnode_id, name, x, y
Nodeinfo2 : endnode_id, name, x, y

Figure 6

programmed to help organize the data, maintain it and even allow users to find exactly what they need in faster and efficient way compared to file-oriented datasets. Databases can also provide important security. They keep all of the information in one place.

For our project, we consider different options such as MySQL, MongoDB and SQLite. MySQL is an open-source relational database management system (RDBMS). MySQL is a popular choice of database for use in web applications. MySQL has poor performance scaling and doesn't perform efficiently when there are too many operations at a given time. On the other hand, MongoDB is a free and open-source cross-platform document-oriented database. Classified as a-No-SQL database, MongoDB avoids the traditional table-based relational database structure in favor of JSON-like documents with dynamic schemas (MongoDB calls the format BSON), making the integration of data in certain types of applications easier and faster. However, the results of MongoDB are inconsistent.

The other option is using SQLite which is a relational database management system contained in a C programming library. In contrast to many other database management systems, SQLite is not a client–server database engine. Rather, it is embedded into the end program. SQLite is a popular choice as embedded database software for local/client storage in application software such as web browsers. It is arguably the most widely deployed database engine, as it is used today by several widespread browsers, operating systems, and embedded systems (such as mobile phones), among others. For the datasets we are using, we store them in SQLite Database. For each dataset, a separate database is built. Each database has two tables, as shown in Figure 6.

B. Inter-process Communication

We currently use network sockets to communicate between server and visualization processes, allowing our system to be deployed across multiple locations on varying hardware configurations. The visualization station sends analytics requests on one socket and receives progress reports and results on another socket. The communication could also be implemented on a single computer with a shared memory object to reduce translation overhead.

C. Sampling Requests

Sampling requests specify the sample algorithm, parameters, and source dataset to the analytics server. The server then spawns a worker thread with the appropriate information. The worker thread stores progress information and the final sampled graph in a shared memory location. During the worker thread's progress, a separate thread sends the percent complete. The sampled graph is used for analytics operations such as CDF computation (covered below). All current samples are saved as GraphML [21] files when the analytics server closes.

D. CDF Computation

Users can also request the sampled graphs be analyzed by their centrality metric distribution (e.g., degree, betweenness, clustering coefficient). As stated previously, igraph provides the implementation for computing these metrics per node. We aggregate the results and render an empirical CDF plot. Ranges of plot values can be specified with a two-finger pinch gesture to highlight the relevant nodes. To facilitate this, we maintain a hash-map container that relates empirical values to lists of node labels. The server can then directly query this map with each (discrete) metric value and build a list of resulting node labels.

Experimental Results

Analytics contributions

First we present illustrative examples of behavior of sampling algorithms. We use four different sampling algorithms as the basic algorithms which includes node sampling, edge sampling, random walk and metropolis hasting. Also we discuss the results of the proposed sampling approaches.

A. Sample Size

One of the most important question that a researcher should ask when planning a study is "How small a sample can be?" If the sample size is too small, even a well-conducted study may fail to answer its research question, may fail to detect important effects or associations, or may estimate those effects or associations too imprecisely. When the size of the sample is small, the

resulting samples can be sparsely connected. Similarly, if the sample size is too large, the study will be more difficult and costly, and may even lead to a loss in accuracy. Hence, optimum sample size is an essential component of any research. Careful consideration of sample size and power analysis during the planning and design stages of clinical research is crucial. We explore how the quality of the sample degrades with the sample size. Figure 7 shows performance of sampling algorithms. We plot the statistic as a function of sample size. As the sample size gets smaller, random walk start to outperform node sampling. For better estimations and results we provide ability to choose a good samples of size based on user preference. In our experiments we used the 5 different sample sizes; 5, 10, 15, 20, 25 percent of original graph.

B. Property Analysis of Basic Algorithms

We run the experiments for sample size 10. Above, we discussed the sample size. In Figure 7, the degree distribution of the sampled graph and original graph is shown. For the basic graph we use, synthetic random graphs including Erdos-Renyi, Barabasi and Watts-Strogatz. As mentioned above, the degree of a node in a network is the number of connections it has to other nodes. The degree distribution is the probability distribution of these degrees over the whole network.

As we can see here, node sampling and Metropolis-Hasting preserve the degree distribution of original graph. Selecting nodes randomly ensures that nodes of a given degree are chosen with probability proportional to the number of such nodes in the graph. Moreover, Metropolis Hasting algorithm converges to uniform sampling of nodes, therefore selected set of nodes have a degree distribution very similar to that of the original graph.

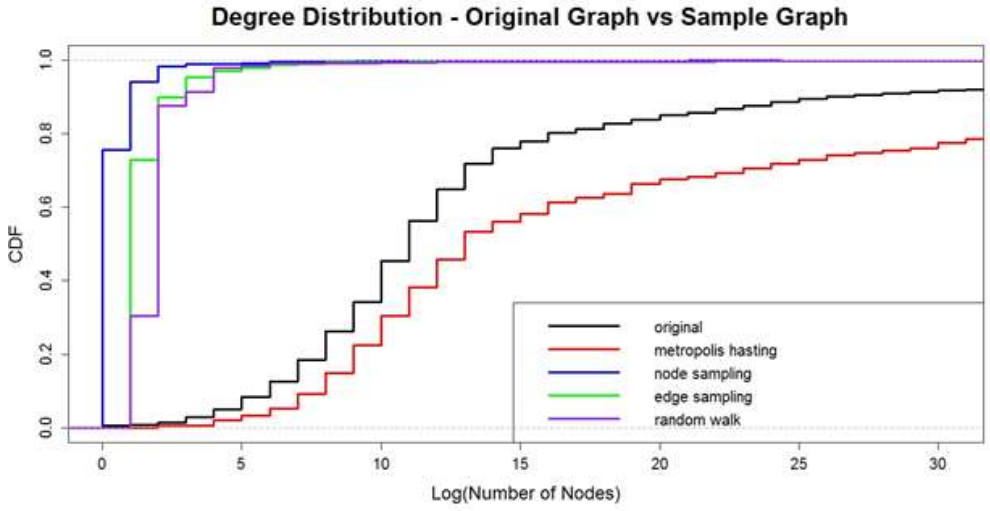

Figure 7. Degree Distribution.

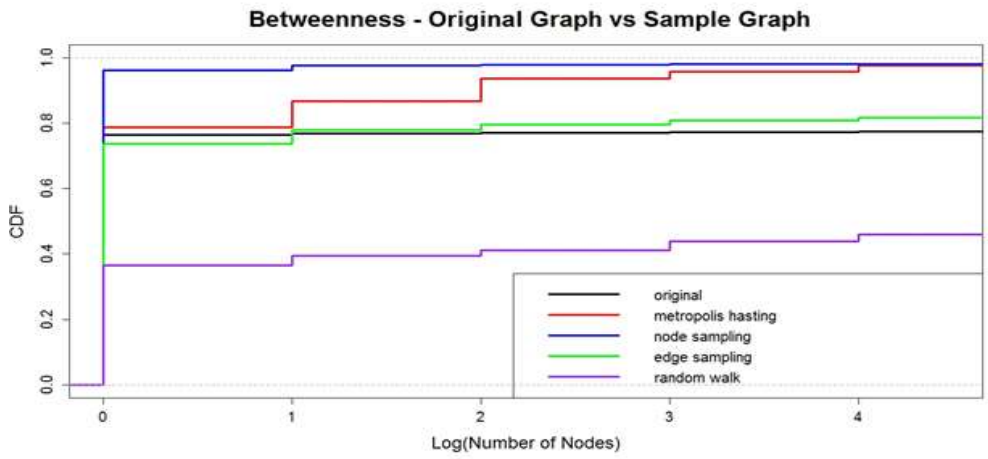

Figure 8. Betweenness Distribution.

The analysis results of betweenness centrality distribution is shown is Figure 8. Centrality measures identify the most important vertices within a graph Betweenness centrality is an indicator of a node's centrality in a network. It is equal to the number of shortest paths from all vertices to all others that pass through that node. A node with high betweenness centrality has a large influence on the transfer of items through the network, under the assumption that item transfer follows the shortest paths. As we can see in this diagram edge sampling preserve the Betweenness of original graph. The reason for that is edge sampling captures path lengths due to its bias toward high-degree nodes and the inclusion of both end points of selected edges.

C. Property Analysis of Proposed Algorithm

Due to the large size and dynamic nature of most of the real-world complex systems, such as peer-to-peer networks, social networks, IP-to-AS mapping networks, etc measuring the quantities of interest on every node is often expensive, while un-baised sampling provides a natural means for estimating system-wide structural characteristic efficiently.

We suggest that our proposed sampling algorithm is a great technique for collecting nearly unbiased samples. We conduct an extensive study by using Human Disease Dataset and Citation Dataset to demonstrate that the proposed approach is an effective one.For evaluation we start the experiments with using three different graphs, (1) Erdos-Renyi, (2) Barabasi, and (3) Watts-Strogatz . Then we analysis the algorithm by using the two real-world datasets, disease dataset and citation graph and plot the patterns.

Figure 9 shows the degree distribution of sampled graphs. The degree distribution of a graph $P(k) = \sum_{k_i} P(k_i, k - k_i)$ is the probability that the degree of is greater than or equal to k. However, when the graph is large, this calculation takes a long time. We can use a sampled

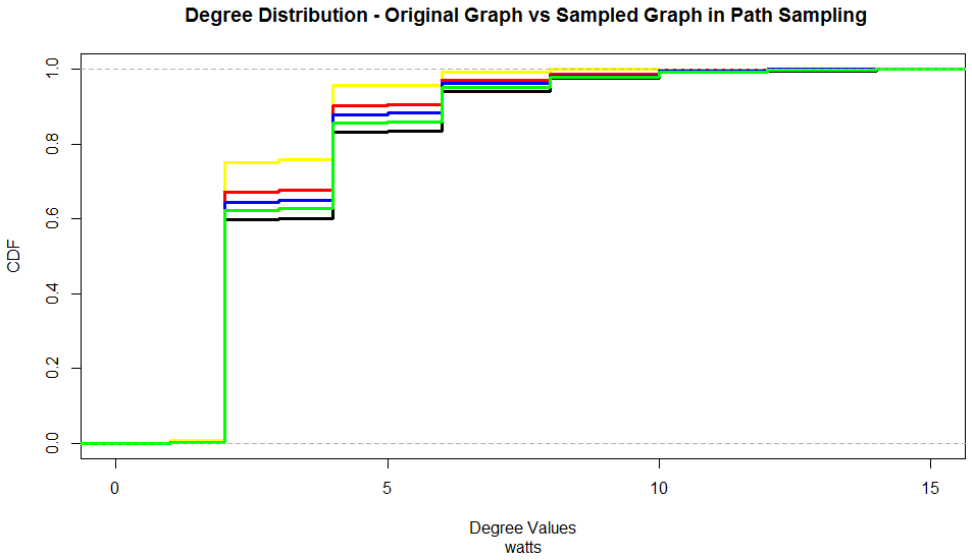

Figure 9. Degree Distribution: Path Sampling

graph with smaller size and estimate the degree distribution of original dataset. As it is shown in Figure 9, the proposed algorithm can estimate the degree distribution of the original graph.

The Figure 10 shows the betweenness distribution of sampled graphs. The complexity of calculating the betweenness distribution for a graph is $(O(|V| \times |E|)$, where |V| and |E| are the number of vertices and edges in the graph. By using a sampled graph which has a smaller size comparing to the original graph, we can estimate the betweenness distribution without processing the large datasets.

D. Visualization Design Goals and Outcomes

Rendering large graphs (e.g., disease diagnosis and citation networks) as node-link diagrams and performing analytics tasks can tax even the most advanced workstation. For real-time interactions, especially touch-based, our design process prioritized timely visual feedback. Our analytics and visualization modules mediate the impact large datasets have on the system's responsiveness. By off-loading the tasks and caching results, our solution prevents a blocking wait between the interactive visualizations and the analytics requests. This also allows the system to be deployed on different computational units: one geared towards analytics and another geared towards visualization. The visualization station stores graph topology in a form better-suited for graphical rendering.

The well-known graphical paradigm of performing scene updates in a draw loop and waiting for a hardware render to complete can also suffer. We have addressed the problems of rendering even a static scene with the large number of graphical objects present in these datasets. When

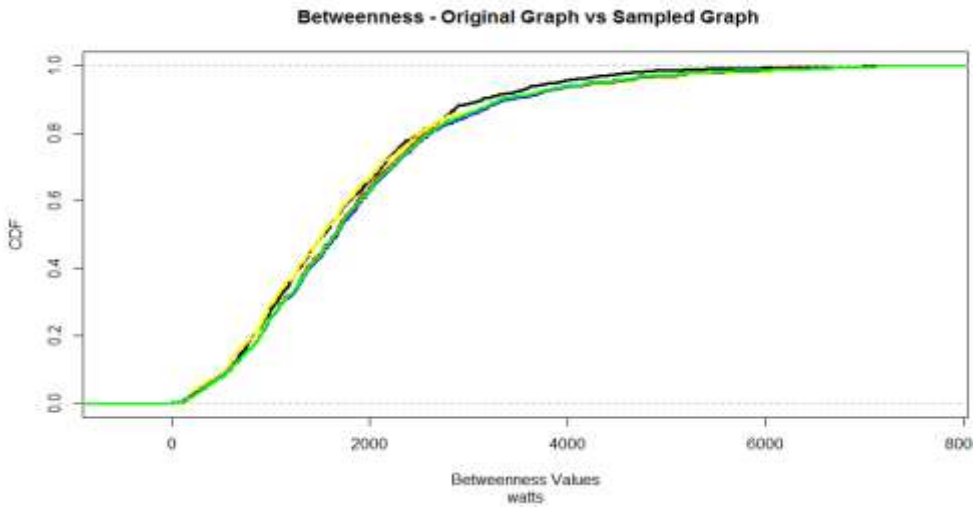

Figure 10. Betweenness Distribution: Path Sampling

graphs are too dense to effectively render directly, we can pre-render to a high resolution texture.

The visualization station also includes functionality to improve the user's role in the analytics process. An interactive sampling mechanism allows users to choose sampling type and parameter (e.g., percentage of edges from a graph). Once created, users can quickly browse through the samples. To analyze the samples, users can request the integrated metric distribution plots and explore nodes within a specified range of metric values. Distribution plots follow the sampling selections and update on parameter changes. Our edge-collapsing summarization creates a coarsened version of the graph based on a density parameter and user navigation gestures.

Evaluation

An interactive system should consider the time taken to produce usable samples. The highly popular Metropolis-Hasting algorithm must investigate many more nodes than are actually selected, due to its degree uniformity requirement. Our path sampling algorithm re-weights edges as sampling continues. Future sampling iterations are guided toward the appropriate nodes and edges giving uniform path sampling.

In sampling large graphs, one of the main goals is to maintain the properties of the original graph. Our proposed unbiased sampling algorithms better maintains the degree distribution for Power Law graphs, compared to the standard sampling technique node sampling. We also maintain the graph's connectivity, as opposed edge sampling, and are not biased towards high degree nodes as with edge sampling and Random Walk. Studies centered on disease outbreak networks require consistency in graph degree and betweenness distributions to investigate the

patterns of disease propagation. Note that betweenness considerations require consistent graph connectivity and help to find communities within the original graph.

By visualizing the data, we can show some aspect of the whole data set so that users can get an impression of a global structure; users can notice outliers or hub nodes. Therefore, they will decide to investigate these outlier nodes further. Our biased graph sampling can help users locate important structural properties. In several applications, such as controlling a disease, accumulating the highest degree nodes into the sample is a great tool. Finding and monitoring these nodes allows predicting the progression and spread of the disease.

We used synthetic networks to experiment the effects of different graph models on the sampling results. We generate random, scale-free, and small world population graphs by using Erdos-Renyi (ER), BarabasiAlbert (BA), and Watts-Strogatz (WS) models, respectively. We observed that Node sampling methods tend to under sample the edges of a population graph. In the other hand, Edge sampling under samples low degree nodes because they have less edges which generates highly disconnected sampled graphs having smaller shortest paths. Random walk based sampling methods require the information on the neighboring nodes be available to the sampling process in order to sample the next step node. Similar to edge sampling, random walk based algorithms tend to over sample high degree vertices compared to low degree ones.

Functionality of Innovation(s)

We are delivering a software system supporting graph sampling and characteristic summarization with graph visualization and touch/gesture support. For a dataset in the GraphML format [21], a SQLite database file can be generated and loaded into our software. A graph sample can then be created, visualized, analyzed, and saved to a GraphML file. The graph sampling algorithms are also provided as a module compatible with igraph. Our experimental evaluations also contribute practical knowledge about information loss in graph sampling. The relationships between sampling methods and graph characteristics were demonstrated for the path sampling, approach and effective uses of sampling for visualization were explored.

Conclusions and Recommendations

In this paper we studied several sampling algorithms. We classified these sampling methods into different groups. Moreover, we discussed how some sampling techniques maintain some properties of the original graphs meanwhile others result in biased sampled graphs. For example, random walk is biased to high degree nodes, while MHRW keeps the degree distribution well and works better in connected graphs. We proposed a new algorithm to achieve an unbiased sampling algorithm. In future we aim to investigate more toward biased

crawling algorithms to achieve the desired samples based on user's demand. Also we are interested to introduce the Simple Random Walk with re-weighting (SRW-rw) and Non-Backtracking Random Walk with re-weighting (NBRW-rw). We are going to extend our work to a new sequential importance of sampling algorithm for generating random graphs with respect to centrality measures such as Eigenvector, Cross-clique and Clustering coefficient.

Based on our findings, we recommend the following priorities for continued development: incorporate other interactive visualization methods and provide users with greater control of the various sampling methods. Users will be able to access summaries and detailed subgraphs to help understand analytics results. One interaction might allow users to specify an area of the sampled graph and request further analysis from the original dataset. Populating the sample with additional information from several random walks or paths could provide greater detail centered on and area of interest.

To help bring the user's attention to these areas of interest, we will perform a highlighting of subgraphs (e.g., common coloring or area shading) based on topology metrics. Currently, we require users to manually adjust sliders on distribution plots and perform time-consuming search tasks. We can instead highlight nodes based on the biases of the selected sampling method or similarities to nodes and subgraphs the user has selected.

Impact and Uses/Benefits

Complex networks have been studied across different fields of science such as mathematical, scientific, and engineering perspectives. For example, a citation network [7] is described as a set of academic papers where a connection represents one paper referencing another. In studying human disease propagation, researchers are interested in finding the correlation between diseases that are often diagnosed together [8]. Diagnoses can form a complex network with diseases connected based on the frequency of co-occurrence in a patient population. Another common example of complex networks can be the World Wide Web, the largest network humanity has ever built. In this network, the nodes are documents and the links are the uniform resource locators (URLs) embedded within the documents.

The proper visualization and monitoring of business processes is crucial for any enterprise. By using unbiased sampling, we can provide an integrated model of the overall process, reduce the size of the dataset, observe connectivity and relationships among business processes, and identify what methods are suitable for analysis. Unbiased graph sampling has enabled scientists to visualize the causal mechanisms behind diseases more easily and extract hidden information from the entities (nodes) and the relations between them (edges). It can facilitate analysis of disease-related molecular data in multiple dimensions.

Alternatively, biased sampling can enable IT administrators to highlight bottleneck processes and their dependencies, improve troubleshooting workflows, and identify important processes running on hosts. However, unbiased sampling provides us with the visibility into how services and applications communicate with each other. We can achieve the knowledge of how a failure can propagate across services. In the medical field, biased sampling has assisted in the discovery of critical diseases and what patient interactions might cause the spread of disease.

For companies and business, it is crucial to retain customers, build loyal relationships and thereby avoid customer acquisition costs. Having a graph of customers and modeling their behavior helps us to have insight of the data and maximize customer loyalty, retention and lifetime value. Unbiased sampling can assist in the analysis and prediction of a new product and its impact on customers. On the other hand, biased sampling can help us to find the core cluster of customers. By understanding the different clusters of customers, we can observer the patterns of firmographic, demographic, and behavioral trends that correlate strongly with high-volume, modestly valuable, and borderline churn customers.

Due to the dynamics of the economy, the marketplace and company decision-making can never be captured fully beforehand. However, by using a sampled model we can estimate potential revenue sources. We can identify which revenue source to pursue, what value to offer, how to price the value, and what markets can bear that price. Unbiased sampling helps us to estimate the impact of a new advertising campaign and estimate revenue grosses. Moreover, by using biased sampling, the managers can visualize and analyze the following factors: sales team productivity, online sales channel productivity variables, and any seasonal changes associated with customer behavior. By reducing the vastness of the commercial data to be analyzed, companies can improve managerial and analyst efficiency, providing a more powerful tool for knowledge workers to interact with large graphs.

List of References

[1] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Reviews of Modern Physics,* vol. 74, no. 1, pp. 47-97, 2002.

[2] R. Cohen and S. Havlin, Complex Networks: Structure, Robustness and Function, Cambridge University Press, 2010.

[3] T. Wang, Y. Chen, Z. Zhang, T. Xu, L. Jin, P. Hui, B. Deng and X. Li, "Understanding Graph Sampling Algorithms for Social Network Analysis," in *31st International Conference on Distributed Computing Systems Workshops*, Minneapolis, MN, USA, 2011.

[4] E. Cem, M. E. Tozal and K. Sarac, "Impact of sampling design in estimation of graph characteristics," in *32nd International Performance Computing and Communications Conference (IPCCC)*, San Diego, CA, 2013.

[5] G. Csardi and T. Nepusz, "The igraph software package for complex network research," *InterJournal,* vol. Complex Systems, p. 1695, 2006.

[6] "Boost C++ Libraries v1.55.0," [Online]. Available: http://www.boost.org. [Accessed 11 November 2013].

[7] J. Gehrke, P. Ginsparg and J. Kleinberg, "Overview of the 2003 KDD Cup," *SIGKDD Explorations,* vol. 5, no. 2, pp. 149-151, 2003.

[8] C. A. Hidalgo, N. Blumm, A.-L. Barabási and N. A. Christakis, "A Dynamic Network Approach for the Study of Human Phenotypes," *PLoS Computational Biology,* vol. 5, no. 4, 2009.

[9] N. K. Ahmed and R. A. Rossi, "Interactive Visual Graph Analytics on the Web," in *9th International AAAI Conference on Web and Social Media*, Oxford, UK, 2015.

[10] R. A. Rossi and N. K. Ahmed, "An Interactive Data Repository with Visual Analytics," *ACM SIGKDD Explorations Newsletter,* vol. 17, no. 2, pp. 37-41, 2015.

[11] A. Ingram, X. Wang and W. Ribarsky, "Towards the Establishment of a Framework for Intuitive Multi-touch Interaction Design," in *International Working Conference on Advanced Visual Interfaces*, Capri Island, Italy, 2012.

[12] D. Baur, B. Lee and S. Carpendale, "TouchWave: Kinetic Multi-touch Manipulation for Hierarchical Stacked Graphs," in *ACM international conference on Interactive tabletops and*

surfaces, Cambridge, MA, USA, 2012.

[13] S. M. Drucker, D. Fisher, R. Sadana, J. Herron and M. Schraefel, "TouchViz: A Case Study Comparing Two Interfaces for Data Analytics on Tablets," in *CHI '13 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Paris, France, 2013.

[14] M. Kurant, A. Markopoulou and P. Thiran, "Towards Unbiased BFS Sampling," *IEEE Journal on Selected Areas in Communications,* vol. 29, no. 9, pp. 1799-1809, 2011.

[15] J. Leskovec and C. Faloutsos, "Sampling from large graphs," in *12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, 2006.

[16] P. Hu and W. C. Lau, "A Survey and Taxonomy of Graph Sampling," *arXiv preprint,* vol. arXiv:1308.5865, 2013.

[17] B. Ribeiro and D. Towsley, "Estimating and sampling graphs with multidimensional random walks," in *10th ACM SIGCOMM conference on Internet measurement*, Melbourne, Australia, 2010.

[18] R. H. Li, J. X. Yu, L. Qin, R. Mao and T. Jin, "On random walk based graph sampling," in *IEEE 31st International Conference on Data Engineering*, Seoul, South Korea, 2015.

[19] "SQLite v3.7.4," [Online]. Available: https://www.sqlite.org. [Accessed 7 December 2010].

[20] "Neo4j Graph Database," [Online]. Available: http://www.neo4j.com. [Accessed 1 January 2015].

[21] "The GraphML File Format," [Online]. Available: http://graphml.graphdrawing.org. [Accessed 1 January 2016].