# Frequent Subgraph Mining on a Single Large Graph Using Sampling Techniques

Ruoyu Zou
Washington State University
Pullman, WA
rzou@eecs.wsu.edu

Lawrence B. Holder
Washington State University
Pullman, WA
holder@wsu.edu

## ABSTRACT
Frequent subgraph mining has always been an important issue in data mining. Several frequent graph mining methods have been developed for mining graph transactions. However, these methods become less usable when the dataset is a single large graph. Also, when the graph is too large to fit in main memory, alternative techniques are necessary to efficiently find frequent subgraphs. We investigate the task of frequent subgraph mining on a single large graph using sampling approaches and find that sampling is a feasible approach for this task. We evaluate different sampling methods and provide a novel sampling method called 'random areas selection sampling', which produces better results than all the current graph sampling approaches with customized parameters.

## Categories and Subject Descriptors
G.2.2 [Graph Theory]: Graph Algorithms, Sampling Algorithms

## General Terms
Theory, Experimentation, Performance

## Keywords
Graph Mining, Sampling, Large Graph.

## 1. INTRODUCTION
Frequent pattern mining has attracted a lot of research in recent years. Many efficient algorithms have been developed for mining frequent itemsets [3, 6, 26, 37], sequential patterns [1, 29], and trees [33, 38]. However, we may be required to find more complicated structures like graphs in some applications. Most works being done on frequent subgraph mining are focused on graph transactions. Examples include gSpan [35], FSG [22]. FFSM [8], MolFea [21], MoSS [5] and Gaston [27]. Holder et al. proposed SUBDUE [12] to discover the best compressing structures. Inokuchi et al. [14] proposed an Apriori based algorithm to discover all frequent substructures. Coatney et al. [7] developed MotifMiner to discover common substructures in Biochemical Molecules. Cong et al. [8] applied frequent substructures discovery in Hierarchical semi-structured data. Bordino et al. introduced large networks mining with subgraph counting in [4]. In many applications, we need to find frequent subgraphs in a single large graph, for instance, discovering structural regularities or anomalies in social network or web structures, which are single graphs and we do not want to split them into parts. However, the algorithms for mining graph transactions cannot be directly used to mine in a single graph even though finding frequent subgraphs in a single graph is more general and applicable [23]. Jiang et al. in [18] try to find globally frequent subgraphs on a single labeled graph. The method that they use is to split the single graph into a set of smaller graphs, and then perform frequent subgraph mining on the smaller graphs. As a result, they are still doing traditional frequent subgraph mining on graph transactions. One drawback of their approach is that they only count once even if more than one instance exists in one smaller graph. Therefore their algorithm will miss some true frequent subgraphs that have many instances in one smaller graph.

One fundamental difficulty in frequent subgraph mining on a single large graph is the size of the graph. Sometimes, the size of a graph can be too large to load into the memory. As a result, can we just use a sample of the single large graph and then perform frequent subgraph mining on the sample graph? Sampling on graphs has been used for some tasks, but not subgraph mining. Leskovec and Faloutsos compared different sampling methods for a single large graph in [25]. They compared the properties of the sample graphs with the original large graph, such as the degree distribution. Jensen et al. analyzed correlation and sampling in relational datasets in [16]. Kashtan et al. proposed an efficient sampling method for estimating subgraph concentration and detecting network motifs in [19]. Kivinen discussed the power of sampling in knowledge discovery in [20]. Lee et al. [24] gave a case in the maintenance of discovered association rules to illustrate the usefulness of sampling in data mining. Toivonen [31] investigated sampling approaches in large datasets for association rules. Zaki [36] evaluated sampling techniques for association rules discovery. However, none of the previous work on graph sampling is related with subgraph mining.

In order to evaluate different sampling approaches, we perform frequent subgraph mining on both the sample graphs and the original large graph, and then compare the results between them. Based on the results, we evaluate the accuracies of different sampling approaches. Beyond that, we also developed a novel sampling method called ‘random areas selection’, which is found to have the highest accuracy among all the current graph sampling methods. We will give more detail on these sampling approaches later. Our main contributions in this paper are as follows:

1. We initiate the work of frequent subgraph mining on a single large graph using sampling approaches.
2. We compare different sampling approaches for frequent subgraph mining on a single large graph.
3. We develop a novel sampling approach for frequent subgraph mining, which is currently the overall best sampling method for the task.


*MLG’10,* July 25–28, 2010 Washington, DC, USA

4. We perform frequent subgraph mining in several domains datasets: Citation Graph, Amazon Graph, and WWW Graph; which are all exceedingly large single graphs.

The rest of the paper is organized as follows. Section 2 describes our formulation of the problem of frequent subgraph mining on a single large graph. Section 3 discusses different sampling approaches for single large graphs and provides more detail on the 'random areas selection' sampling approach. Section 4 describes the experimental evaluation on the Citation Graph, Amazon Graph and WWW Graph. Section 5 presents conclusions and future work.

## 2. PROBLEM DEFINITION

In this paper, we use labeled graphs. To make the description of this paper clearer, we give the following definitions.

DEFINITION 1: ***Labeled Graph:*** A labeled graph can be represented by a 4-tuple, G = (V, E, L, l), where

V is a set of vertices (or nodes)

$E \subseteq V \times V$ is a set of edges, they can be directed or undirected

L is a set of labels

$l: V \cup E \rightarrow L$, l is a function assigning labels to the vertices and the edges

There can be multiple edges between two vertices, however no self-edge is allowed in our experiment.

### 2.1 How to define the support of a subgraph

A fundamental issue that needs to be addressed for subgraph mining is the counting method used for occurrence frequency. When the input is a database of graphs, the support (or frequency) of a subgraph is the number of graphs in the given database that contain the subgraph. However, if the input is a single labeled graph, the definition needs to be changed. In general, there are two possible counting methods. In the first method, two occurrences are different as long as they have at least one different edge. Therefore, overlaps of occurrences of the same subgraphs are allowed. In the second method, two occurrences are considered different if they do not share any edges, which means that no overlap is allowed. However, sharing nodes is allowed. This is illustrated in figure 1. If we allow overlaps, s1 and s2 can be counted as two occurrences of the same subgraph, while s1 and s2 can be counted as only one occurrence if we do not allow overlaps.

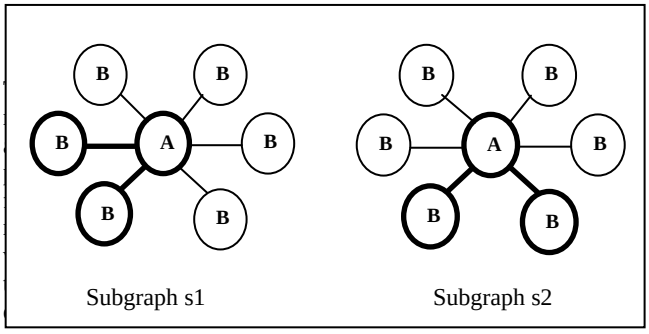


**Figure 1: Two overlapping instances of the subgraph B-A-B.**

One important property for frequent subgraph mining is the downward closure property, which is extensively used to prune the search space. If we use the first method for support definition, the downward closure property no longer holds. For example, in figure 2, while s2 is a super graph of s1 ($s1 \subseteq s2$), s2 has a higher frequency than that of s1 in the input graph G. Therefore, a subgraph mining algorithm cannot utilize this property to prune the search space, and it needs to search the whole candidate space to find all the frequent subgraphs. This makes the discovery process extremely time consuming. On the other hand, if we are using the second method that does not allow overlaps, then the frequency counting is downward closed, and we can use this property to prune candidate space. There is another definition of support proposed by M. Fiedler and C. Borgelt in [10], in which the definition can maintain the downward closure property while still allowing some kinds of overlaps. However, their definition for support only counts once if two subgraphs are extended from the same ancestor, which in some cases we need to count it as twice.

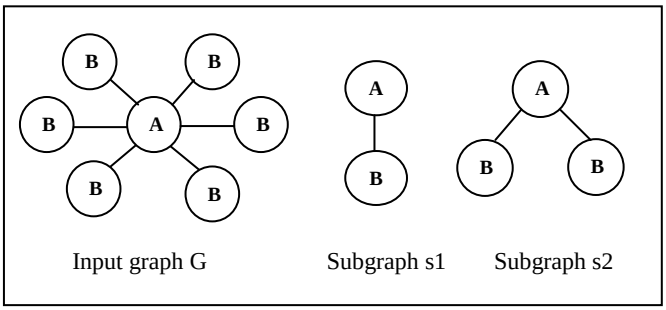


**Figure 2: Scenario in which s1 is more frequent without overlap, but s2 is more frequent with overlap.**

In order to utilize the downward closure property for the mining process, we adopt the second method for frequency counting in this paper. Given this definition, we formulate the frequent subgraph mining problem as follows:

DEFINITION 2: ***Graph Isomorphism:*** An isomorphism is a bijective function $f: V(G) \rightarrow V(G')$, such that

$\forall u \in V(G), l_G(u) = l_{G'}(f(u))$

$\forall (u, v) \in E(G), (f(u), f(v)) \in E(G')$ and
$l_G(u, v) = l_{G'}(f(u), f(v))$

There is a subgraph isomorphism from G to $G'$ if there is a subgraph of $G'$ that is isomorphic to G.

DEFINITION 3: ***Frequent Subgraph Mining on a Single Large Graph:*** Given a graph G, and a minimum support minSup, let $\sigma(g, G)$ denote the occurrence frequency of g in G, i.e, the number of non-overlapping subgraph isomorphisms of g in G. Frequent subgraph mining on a single large graph is to find every subgraph g of G, such that $\sigma(g, G)$ is greater than or equal to minSup.

DEFINITION 4: ***Induced Subgraph:*** A subgraph g is said to be an induced subgraph of G if, for any pair of vertices $v_i$ and $v_j$ of g, $v_i v_j$ is an edge of g if and only if $v_i v_j$ is an edge of G.

This definition will be used in random nodes sampling. The sample graph is the induced graph from the sample nodes.

DEFINITION 5: ***Sample:*** Given a labeled graph G = (V, E, L, l), a graph S = (V′, E′, L′, l′) is a sample from G, where

$V' \subseteq V, E' \subseteq E, L' \subseteq L$ and $l' \subseteq l$.

One sample of a big graph is different from one subgraph of a big graph. One sample of a big graph is a representative of the big graph and it can be disconnected, while one subgraph of a big graph is not necessarily a representative of a big graph and is typically connected.

# 3. DIFFERENT SAMPLING METHODS

In graph sampling, given a large target graph, our task is to extract a smaller graph possessing certain properties of interest that are representative of the target graph. Traditional techniques for evaluating a sampling method compare global metrics of the sample to the same global metrics of the target graph (e.g., average degree). In contrast we are interested in comparing the frequent subgraphs found in the sample graph to those found in the target graph.

We investigate four different sampling methods for the frequent subgraph mining task: random nodes selection sampling, random edges selection sampling, random walk sampling, and random areas selection sampling. Pseudo code for each approach is given with their descriptions. We evaluate these sampling algorithms using three real world datasets. Descriptions of the datasets will be given later. We will see in section 4 that random areas selection sampling produces the overall best performance.

## 3.1 Random Nodes Selection Sampling

One way to create a sample graph is to randomly select a set of nodes N with uniform probabilities, and then the sample graph is the subgraph induced by the nodes in N. We call this sampling method *random nodes selection sampling*. Algorithm 1 shows the pseudocode for this sampling method.

**Algorithm 1: Random nodes selection sampling**

Input: Graph G, sample size S (number of nodes)
$S_G$: Sample graph
N: number of nodes currently in $S_G$
V(G) : All of the vertices in G
$S_G$ = {}
N = 0
While (N < S)
  Randomly select one vertex n from V(G)
  If (n is not in $S_G$)
    Insert n to $S_G$
  N = N + 1
For each pair of nodes (u, v) in $S_G$
  If (there is an edge between u and v in G )
    Add an edge for these two vertices in $S_G$
Return $S_G$

Stumpf et al. in [30] showed that the random nodes selection sampling approach does not preserve the power law distribution property. Leskovec et al. in [25] investigated the degree distribution of sample graphs using random nodes selection sampling. They showed that the random nodes selection sampling method is good at preserving degree distributions. Here we are interested in whether the frequent subgraphs in the target graph are still frequent in the sample graph. One variant to the random nodes selection sampling method is to use a non-uniform probability distribution over the nodes in the target graph. A common way for doing this is to set the probability of a node being selected for the sample graph to be proportional to its degree [25]. This method prefers high degree nodes, and hence produces too dense graphs.

## 3.2 Random Edges Selection Sampling

Random edges selection sampling randomly selects a set of edges from the target graph according to a uniform probability distribution over the edges. The sample graph consists of the set of edges and the vertices incident to these edges. Algorithm 2 shows the pseudocode for this sampling method.

**Algorithm 2: Random edges selection sampling**

Input: graph G, sample size S (number of edges)
$S_G$: Sample graph
N: number of edges currently in $S_G$
E(G): All of the edges in G
$S_G$ = {}
N = 0
While (N < S)
  Randomly select one edge e from E(G)
  If (e is not in SG)
    Insert e into SG
  N = N + 1
Return $S_G$

There are some problems with this algorithm. First, the sampled graph may be sparsely connected (even disconnected) and will have large diameters. Therefore it is not good at finding small diameter subgraphs. Second, this sampling method is not good at preserving structure information. This is shown in the experiments that the sampled graphs are not preserving frequent subgraphs. A variation of this approach is the following. First, randomly pick a node with uniform probability, and then randomly pick an edge incident to that node with uniform probability. This method can avoid generating sparse graphs; however it is biased to high degree nodes because they have more edges connecting to them.

## 3.3 Random Walk Sampling

For random walk sampling, given the number of nodes S to be in the sample graph, we first randomly pick a starting node and then perform a random walk on the graph. At every step, we randomly choose one node adjacent to the previously chosen node. To avoid selecting duplicate nodes, we do not allow returning to the most recent previous node. If the process gets stuck during the walking process, we randomly select another starting node which is different from the current set of selected nodes. Continuing from the new starting node, we perform random walk again until the number of nodes reaches sample size S. The sample graph is the induced subgraph from the set of sampled nodes. For a highly-connected graph, this method will likely produce a small number of disconnected graphs, which will put it at a disadvantage for frequent subgraph mining if the subgraph size is large. Algorithm 3 shows the pseudocode for this sampling method.

**Algorithm 3: Random walk sampling**

Input: graph G, sample size S
$S_G$: Sample graph
N: number of nodes currently in $S_G$
N = 1
$S_G$ = {v | v is a random vertex from G}
While (N < S)
  Neighbors = vertices adjacent to v not in $S_G$
  If (Neighbors empty)
    v=random vertex from G
  Else
    v=random selected vertex from Neighbors
  If v not in $S_G$
    Add v to $S_G$
    N = N +1
For each pair of nodes (u, v) in $S_G$
  If (there is an edge between u and v in G)
    Add an edge for these two vertices into $S_G$
Return $S_G$

## 3.4 Random Areas Selection Sampling

Unlike all of the above sampling approaches that select one node (one edge) at a time, the random areas selection sampling method picks one area of the graph each time. Given a sample size S, N is the set of nodes in the sample and A is the number of areas in the sample. Initially, N is empty. We first randomly select A starting nodes and add them to N. Then we find all the nodes in G adjacent to nodes in N and add them into N. We repeat this process until the sample size reaches S. Algorithm 4 shows the pseudocode for this sampling method.

**Algorithm 4: Random areas selection sampling**

Input: graph G, sample size S, number of areas A
$S_G$: Sample graph
N: Number of nodes currently in $S_G$
$V_S$: set of nodes currently in $S_G$
$S_G$ = {}
$V_S$ = {}
Randomly select unique nodes $n_0, n_1 \ldots n_N$ as the starting nodes

$V_S = \{n_0, n_1 \ldots n_N\}$
N= A
While (N < S)
  Neighbors = all nodes adjacent to nodes in $V_S$
  $V_S = V_S$ + Neighbors
  N = N + |Neighbors|
$S_G$ = graph induced by $V_S$ from G
Return $S_G$

Since the last set of neighbors adding to the sample is very large, we will stop adding nodes into the sample after N reaches the sample size S. Therefore it is possible that only part of the neighbors in the last set will be added to the sample.

Figure 3 is an example of the random areas selection sampling method. Given the sample size 14 and number of areas 2, we can generate sampled graphs s1 and s2 in the circles. From the figure, we can see that the random areas selection sampling method is better at preserving structure information. Note that the performance of this method is sensitive to the number of areas. If the frequent subgraph sizes are small, choosing a larger number of areas can help to capture more frequent subgraphs in the sample graph. On the other hand, if the frequent subgraph sizes are relatively large, choosing a smaller number of areas can help to preserve better structure information, because each area has a larger size and hence more likely to contain large subgraphs. Also, increasing the sample size can help finding large subgraphs.

# 4. EXPERIMENTAL EVALUATION

In this section we present experimental results on three real graph datasets to evaluate the performance of the different sampling methods for the task of frequent subgraph mining on single large graphs.

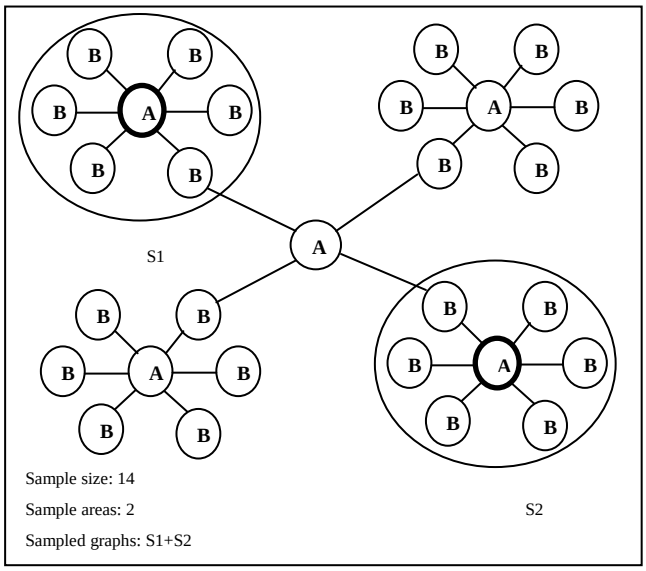


**Figure 3: Random Areas Selection Sampling**

## 4.1 Datasets Description

We experimented on three different datasets: Citation Graph, Amazon Transaction Graph, and WWW Graph. These datasets are available at www.cs.yale.edu/homes/mmahoney/NetworkData. We developed a frequent subgraph discovery on a single large graph system (FSGS) based on SUBDUE [12] for the experimental evaluation. We use FSGS to get the 10 most frequent subgraphs for each dataset. Here we choose the 10 most frequent subgraphs for two reasons. First, frequent subgraph mining on a single graph is substantially more time consuming than frequent subgraph mining on graph transactions. Second, the datasets that we are using are very large. As a result, in order to allow the experiments to complete in reasonable time, we choose the 10 most frequent subgraphs. Additionally, to make the frequent subgraphs that we find more applicable, we set the minimal size of frequent subgraphs to be 4 so that we can find larger frequent subgraphs. Since the original sizes for Amazon graph and WWW graph are extremely large, we reduced the size of Amazon Graph and the WWW graph in order to find the true frequent subgraphs in a reasonable amount of time. The method

we use to truncate the datasets is to randomly delete some nodes and edges with equal probabilities in order to avoid human bias for the deletion process. Nonetheless, these reduced graphs are still very large. The only reason that we want to reduce the size of the graphs is to make them smaller. The reducing size process will not affect the performance of different sampling methods. The number of nodes and number edges for each dataset are given in Table 1. Note that all the nodes and edges have the same label in the three graphs (i.e., unlabeled).

**Table 1: Datasets description**

| Dataset | Number of Nodes | Number of Edges |
|---|---|---|
| Citation Graph | 27400 | 352021 |
| Amazon Graph | 38750 | 490320 |
| WWW Graph | 39200 | 503208 |

The citation graph comes from the 2003 KDD Cup. The vertices of the graph are different papers. The edges of the graph are the citation relationships between different papers. An edge will be added between two paper vertices if one paper cites the other paper.

The Amazon graph is collected using transactions data from the Amazon website in 2003. The vertices represent different transaction parties. If one party has a transaction with another party, an edge is added between the two parties.

The WWW graph comes from different web pages and the links between them on the internet. Each vertex represents one web page. The edges represent the links between different web pages.

## 4.2 Experimental Setup

We first run FSGS on the original three datasets to get the 10 most frequent subgraphs for each dataset. The frequent subgraphs that we get on this step are the true frequent subgraphs. Then we run the different sampling methods to get sampled graphs from the target graphs. To reduce sampling bias, we run each sampling algorithm N times to get N sample graphs instead of one sample graph for each sampling algorithm. In our experiment, we set the N to be 15 in order to make the experiment finish in reasonable time. After that, FSGS performs frequent subgraph mining on the sample graphs. Next we compare the frequent subgraphs from the sample graphs and frequent subgraphs from the original graphs. The next section describes our approach for comparing the sets of frequent subgraphs from the original and sample graphs.

## 4.3 Evaluation Metric

First, we give an example for how to evaluate different sampling approaches. FSGS can output the frequent subgraphs and number of instances for each frequent subgraph. For example, after performing frequent subgraph mining on the citation graph G, FSGS gets the 10 most frequent subgraphs $G_1$, $G_2$, $G_3$ …, and $G_{10}$. The numbers of instances for $G_1$, $G_2$, $G_3$ …, and $G_{10}$ are $N_1$, $N_2$, $N_3$ …, $N_{10}$ respectively. Note that $G_1$, $G_2$, $G_3$ …, and $G_{10}$ here are the true frequent subgraphs for the citation graph G. To get the sample graphs from the citation graph G, we run one of the sampling methods on G to get 15 different sample graphs $S_1$, $S_2$, $S_3$ …, and $S_{15}$. Then we perform frequent subgraph mining on $S_1$, $S_2$, $S_3$ …, and $S_{15}$ to get the 10 most frequent subgraphs and their corresponding numbers of instances for each of the 15 sample graphs. Below are the frequent subgraphs and their corresponding instance numbers for each sampled graph,

- $S_1$: $<G_{s1}^1, N_{s1}^1>, <G_{s1}^2, N_{s1}^2>$ $<G_{s1}^3, N_{s1}^3>, <G_{s1}^4, N_{s1}^4>$ $<G_{s1}^5, N_{s1}^5>, <G_{s1}^6, N_{s1}^6>$ $<G_{s1}^7, N_{s1}^7>, <G_{s1}^8, N_{s1}^8>$ $<G_{s1}^9, N_{s1}^9>, <G_{s1}^{10}, N_{s1}^{10}>$

- $S_2$: $<G_{s2}^1, N_{s2}^1>, <G_{s2}^2, N_{s2}^2>$ $<G_{s2}^3, N_{s2}^3>, <G_{s2}^4, N_{s2}^4>$ $<G_{s2}^5, N_{s2}^5>, <G_{s2}^6, N_{s2}^6>$ $<G_{s2}^7, N_{s2}^7>, <G_{s2}^8, N_{s2}^8>$ $<G_{s2}^9, N_{s2}^9>, <G_{s2}^{10}, N_{s2}^{10}>$

…

…

…

- $S_{15}$: $<G_{s15}^1, N_{s15}^1>, <G_{s15}^2, N_{s15}^2>$ $<G_{s15}^3, N_{s15}^3>, <G_{s15}^4, N_{s15}^4>$ $<G_{s15}^5, N_{s15}^5>, <G_{s15}^6, N_{s15}^6>$ $<G_{s15}^7, N_{s15}^7>, <G_{s15}^8, N_{s15}^8>$ $<G_{s15}^9, N_{s15}^9>, <G_{s15}^{10}, N_{s15}^{10}>$

*$G_{si}^j$ is the jth most frequent subgraph in sample $S_i$*
*$N_{si}^j$ is the corresponding number of instances for $G_{si}^j$*

We compare, using graph isomorphism, the frequent subgraphs $G_1$, $G_2$, $G_3$ …, and $G_{10}$ from the original graph G and the frequent subgraphs $<G_{s1}^1, G_{s1}^2, G_{s1}^3, G_{s1}^4, G_{s1}^5, G_{s1}^6, G_{s1}^7, G_{s1}^8, G_{s1}^9, G_{s1}^{10}>$, $<G_{s2}^1, G_{s2}^2, G_{s2}^3, G_{s2}^4, G_{s2}^5, G_{s2}^6, G_{s2}^7, G_{s2}^8, G_{s2}^9, G_{s2}^{10}>$ …, and $<G_{s15}^1, G_{s15}^2, G_{s15}^3, G_{s15}^4, G_{s15}^5, G_{s15}^6, G_{s15}^7, G_{s15}^8, G_{s15}^9, G_{s15}^{10}>$ from the 15 sample graphs to get the common frequent subgraphs set between them. For example, if $<G_1, G_3, G_7>$ is the same as $<G_{s1}^1, G_{s1}^2, G_{s1}^3>$ between $<G_1, G_2, G_3$ …, and $G_{10}>$ and $<G_{s1}^1, G_{s1}^2, G_{s1}^3$ … and $G_{s1}^{10}>$, the common frequent subgraphs set between G and $S_1$ is $<G_1, G_3, G_7>$.

The accuracy of the sampling method is defined as:

$$\text{Accuracy} = \frac{1}{\mathrm{T}} * \sum_{\mathrm{i=1}}^{\mathrm{i=T,}} \text{Instances}(\mathrm{C_i}) \,/\, \sum_{\mathrm{k=1}}^{\mathrm{k=F}} \mathrm{N_k}$$

This matrix is used to represent the percentage of frequent subgraphs in the original big graph that are also frequent in the small sample graph.

T is the number of samples, i.e., how many samples we extract from the input graph. We set it to 15 for our experiments.

F is the number of frequent subgraphs that we extract from each graph. In our experiments, we extract the 10 most frequent subgraphs. $N_k$ is the number of instances of the $k^{th}$ frequent subgraph in the original graph G.

$C_i$ are the common frequent subgraphs between $G_i$ and $G_{si}^j$. For example, if $<G_1, G_2, G_4, G_5, G_7, G_8, G_{10}>$ exist in both $<G_1, G_2, G_3, G_4, G_5, G_6, G_7, G_8, G_9, G_{10}>$ and $<G_{s1}^1, G_{s1}^2, G_{s1}^3, G_{s1}^4, G_{s1}^5, G_{s1}^6, G_{s1}^7, G_{s1}^8, G_{s1}^9, G_{s1}^{10}>$, the value of $C_1$ is $<G_1, G_2, G_4, G_5, G_7, G_8, G_{10}>$, which is the set of common subgraphs between G and $S_1$.

Instances ($C_i$) is the sum of all the numbers of instances of graphs in $C_i$ in sample graph $i$. For example, if $C_1$ is <$G_1$, $G_2$, $G_4$, $G_5$, $G_7$, $G_8$, $G_{10}$> and the corresponding numbers of instances for $C_1$ is <$N_1$, $N_2$, $N_4$, $N_5$, $N_7$, $N_8$, $N_{10}$>, the value of Instances($C_i$) is $N_1 + N_2 + N_4 + N_5 + N_7 + N_8 + N_{10}$.

## 4.4 Results

We compare the accuracies of the different sampling methods based on the three datasets: Citation Graph, Amazon Transaction Graph and WWW Graph. The results are shown in figure 3, figure 4 and figure 5. Here we vary the sample size as a percentage of the size of the entire graph. For the random areas selection sampling approach, the number of areas is set to be 10. We can see that the random areas selection sampling method gives the best overall performance for all of the three datasets.

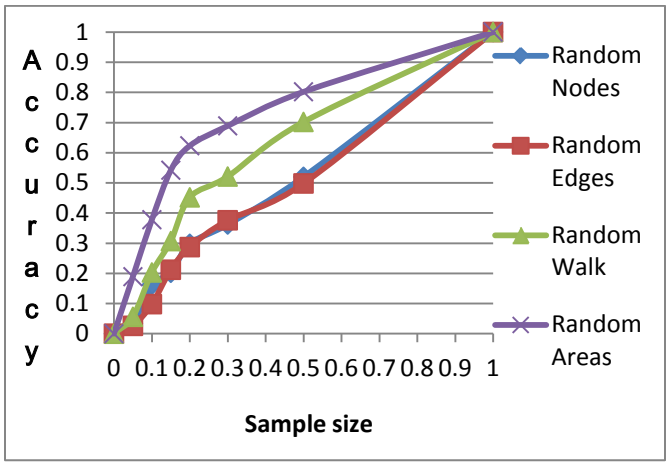


**Figure 3: Citation Graph**

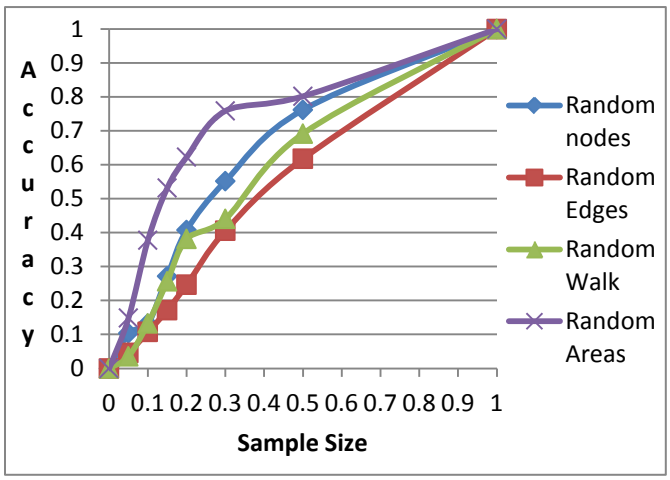


**Figure 4: Amazon Transaction Graph**

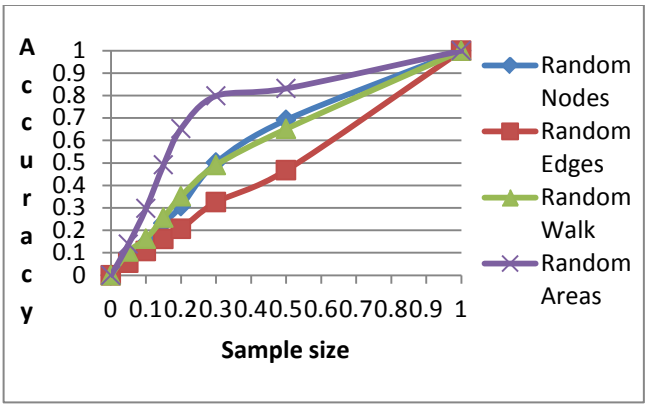


**Figure 5: WWW Graph**

The random nodes sampling and random edges sampling methods tend to find small subgraphs. The random walk sampling method has a strong bias toward chains. For more complex substructures, the random areas selection sampling method works better than all of the above methods. Unfortunately, we can see that the accuracy does not plateau before the sample size reaches the size of the entire original graph for all the sample methods, which indicates that sampling approach is not capable of finding all the true frequents subgraphs. One reason for this is because our evaluation matrix takes the sizes of subgraphs into account. Subgraphs are assigned more weights if their sizes are larger. Therefore, when the sample size is small, it is difficult to capture large size subgraphs and resulting in smaller evaluation matrix value. However, if we want an approximation of frequent subgraphs, sampling is still a viable method. Another way we will do in the future is using inexact matches and corresponding weights for each inexact match, which might improve the accuracy measure.

One important property that we find for the random areas sampling method is that the number of areas that we choose plays a key role in the performance. We illustrate this property on the three datasets by varying the number of areas. The sample size is 10 percent of the original graph. The results are shown in figure 6, figure 7 and figure 8.

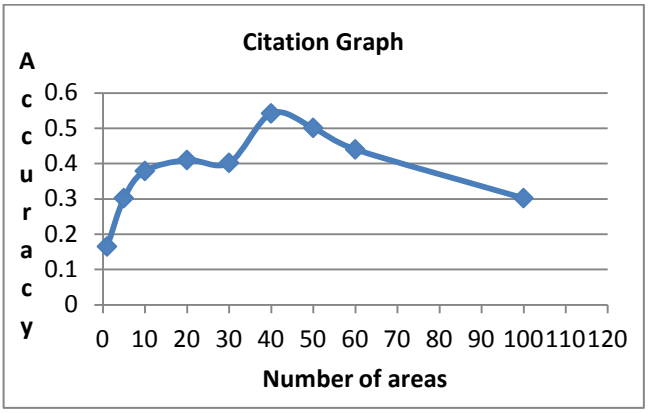


**Figure 6: Different number of areas for Citation Graph**

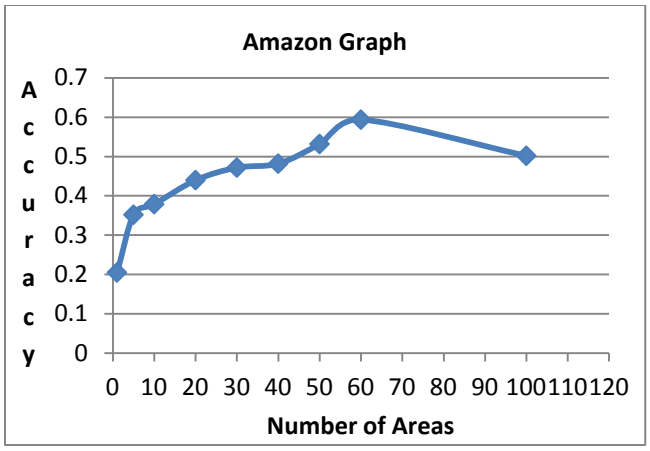


**Figure 7: Different number of areas from Amazon Graph**

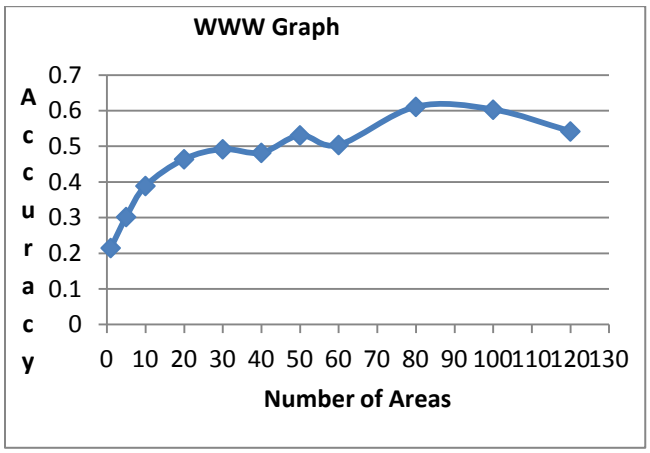


**Figure 8: Different number of areas for WWW Graph**

From Figures 6, 7 and 8, we can see that there are many local maxima accuracy values based on different numbers of areas. Therefore, it is difficult to determine how many areas to choose in order to get a global maximum for the accuracy. Based on our experiments, this number is greatly dependent on the properties of the original large graph that we need to sample from.

## 5. Boundary Analysis

For the random areas selection sampling method, a question to ask is how many nodes we need to select so that we can guarantee the frequent subgraphs from the sample are the frequent subgraphs from the original graph. If we are performing frequent subgraph discovery on graph transactions, the sample size can be determined by applying Chernoff Bounds. Zaki et al. addressed this problem in [36].

Let $\tau$ denote the support of subgraph I. We want to select n sample transactions out of a total of N transactions in dataset D. The variable X gives the number of transactions in the sample containing I. For any positive constant, $0 \leq \varepsilon \leq 1$, the Chernoff Bounds state that

$$P(X \leq (1-\varepsilon)n\tau) \leq e^{-\varepsilon^2 n\tau/2} \quad (1)$$

$$P(X \geq (1+\varepsilon)n\tau) \leq e^{-\epsilon^2 n\tau/3} \quad (2)$$

The Chernoff Bounds provide the accuracy of the sample, which is given by 1-$\epsilon$. It also gives us the confidence value that the sample size n will have a given accuracy. Equation (1) gives the lower bound of the confidence value and equation (2) gives the upper bound of the confidence value.

From equation (1) and (2), we can determine the sample size n for accuracy $1-\varepsilon$ and confidence 1-c. The sample size is

$$n = -2\ln(c)/(\tau\varepsilon^2) \quad (3)$$

The above formulas are used for sampling on graph transaction. To apply them for random areas selection sampling on a single graph, we can view different areas as independent graph transactions since different areas are not overlapping with each other. As a result, we can model the problem as follows.

Let $S'$ denote the size of the sample graph, and A is the number of areas in the sample graph, and the size for each area is $S_A$. S is the size of the original graph. As a result, the total number of areas in the original graph is $/S_A$ . Let $\tau$ denote the support of subgraph I. We want to select A number of areas out of $SA/S'$ number of areas. The variable X gives the number of instances of I in the sample graph. For any positive constant, $0 \leq \varepsilon \leq 1$, the Chernoff Bounds state that

$$P(X \leq (1-\varepsilon)A\tau) \leq e^{-\varepsilon^2 A\tau/2} \quad (4)$$

$$P(X \geq (1+\varepsilon)A\tau) \leq e^{-\epsilon^2 A\tau/3} \quad (5)$$

From (4) and (5), we can determine the number of areas A in the sample graph in order to get accuracy $1-\varepsilon$ and confidence 1-c.

$$A = -2\ln(c)/(\tau\varepsilon^2) \quad (6)$$

Since the size for each area is $S_A$, the size $S'$ of the sample graph is

$$S' = -2S_A\ln(c)/(\tau\varepsilon^2) \quad (7)$$

To apply formula (7) on the citation graph, we set the size for each area as one percent of the original graph. Therefore, $S_A = 1\% * 27400 = 274$. Let accuracy $1-\varepsilon = 0.9$ and confidence 1-c = 0.9, i.e., $\varepsilon = 0.1$ and c = 0.1. The support $\tau$ is 10. The size $S'$ of the sample graph is

$$S' = -2 * 274 * \frac{\ln(0.1)}{10*0.1^2} = 12618$$

According to this analysis, the sample size should be almost half the size of the original graph. Clearly, our sampling scenario violates the assumptions underlying the Chernoff bounds, but the results provide an estimate of the sample size needed to guarantee success under the above assumptions. We will pursue a more appropriate bounds model in future work.

## 6. CONCLUSION AND FUTURE WORK

We have presented an experimental evaluation of different sampling approaches in order to perform efficient and accurate frequent subgraph mining on a single large graph. The experimental results indicate that the "random areas selection" method that we propose provides the overall best performance. For future work we intend to develop a more accurate framework for deriving a bound on the sample size of the "random areas selection" approach necessary to find the frequent subgraphs with some level of confidence. We will also work on how to

automatically decide the number of areas for the "random areas selection" sampling approach. The datasets we are using now contain same label information. We will run our method on graphs with more diversity of labels, which will likely decrease the running time.

## 7. REFERENCES


[1] R. Agrawal and R. Srikant. *Mining Sequential Patterns. In ICDE*, pp: 3-14, 1995.

[2] T. Asai, K. Abe, S. Kawasoe, H. Arimura, H.Satamoto, and S. Arkiwa. *Efficient Substructure Discovery from Large Semi-Structured Data*. In SIAM, 2002.

[3] R. J. Bayardo. *Efficiently Mining Long Patterns from Databases*. In ACM SIGMOD, pp:85-93, 1998.

[4] I. Bordino, D. Donato, A. Gionis, S. Leonardi. Mining Large Networks with Subgraph Counting. ICDM, pp: 737-742, 2008.

[5] C. Borgelt and M.R. Berthold. Mining Molecular Fragments: Finding Relevant Substructures of Molecules. In ICDM, pp: 51-58, 2002.

[6] D. Burdick, M. Calimlim, and J. Gehrke. *MAFIA: A Maximal Frequent Itemset Algorithm for Transactional Databases*. In ICDE, pp: 443-452, 2001.

[7] M. Coatney, S. Parthasarathy. *MotifMiner: Efficient Discovery of Common Substructures in Biochemical Molecules.* Knowledge and Information Systems, 7(2) pp: 202-223, 2005.

[8] C. Cong, L. Yi, B. Liu and K. Wang. *Discovering Frequent Substructures from Hierarchical Semi-structured Data* In SDM, pp: 175-192, 2002.

[9] *The D Statistic.* http://mste.illinois.edu/patel/chisquare/dstat.html.

[10] M. Fiedler and C. Borgelt. *Support Computation for Mining Frequent Subgraphs in a Single Graph*. http: //mlg07.dsi.unifi.it/pdf/40_Borgelt.pdf.

[11] P. Hintsanen and H. Toivonen. *Finding reliable subgraphs from large probabilistic graphs.* In DMKD, 17(1) pp: 3–23, 2008.

[12] L. Holder, D. Cook and S. Djoko. *Substructure Discovery in the Subdue System.* In AAAI Workshop on Knowledge Discovery in Databases, pp: 169–180, 1994.

[13] J. Huan, W. Wang, J. Prins. Efficient Mining of Frequent Subgraphs in the Presence of Isomorphism. In ICDM, 2003.

[14] A. Inokuchi, T. Washio and H. Motoda. *An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data.* In PKDD, pp: 13-23, 2000.

[15] S. Jacquemont, F. Jacquenet and M. Sebban. *A Lower Bound on the Sample Size Needed to Perform a Significant frequent Pattern Mining Task.* In Pattern Recognition Letters, 30(11) pp: 960–967, 2009.

[16] D. Jensen and J. Neville. *Correlation and Sampling in Relational Data Mining.* In Proceedings of the 33rd Symposium on the Interface of Computing Science and Statistics, 2001.

[17] J. Han, J. Pei, and Y. Yin. *Mining Frequent Patterns without Candidate Generation.* In SIGMOD, pp: 1-21, 2000.

[18] X. Jiang, H. Xiong, C. Wang, and A. Tan, *Mining Globally Distributed Frequent Subgraphs in a Single Labeled Graph*, In Data & Knowledge Discovery, 68(10), pp: 1034-1058.

[19] N. Kashtan, S. Itzkovitz, R. Milo and U. Alon. *Efficient Sampling algorithm for estimating subgraph concentrations and detecting network motifs*. In Bioinformatics, 20(11), pp: 1746-1758, 2004.

[20] J. Kivinen and H. Mannila. *The power of Sampling in Knowledge Discovery.* Proceedings of the thirteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems, pp: 77–85, 1994.

[21] S. Kramer, L. de Raedt, and C. Helma. Molecular Feature Mining in HIV Data. In ACM SIGKDD, pp: 136-143, 2001.

[22] M. Kuramochi and G. Karypis. *Frequent Subgraph Discovery.* In ICDM, pp: 313-320, 2001.

[23] M. Kuramochi and G. Karypis. *Finding Frequent Patterns in a Large Sparse Graph.* In Data Mining and Knowledge Discovery, 11(3) pp: 243-271, 2005.

[24] S.D. Lee, D. Cheung and B. Kao. *Is Sampling Useful in Data Mining? A Case in the Maintenance of Discovered Association Rules.* In Data Mining and Knowledge Discovery, pp: 233-262, Sep, 1998.

[25] J. Leskovec and C. Faloutsos. *Sampling from large Graphs.* In SIGKDD, PP: 631–636, 2006.

[26] H. Mannila, H. Toivonen, and A. I. VerKamo. *Discovery of Frequent Episodes in Event Sequences.* In Data Mining and Knowledge Discovery, pp: 259-289, 1997.

[27] S. Nijssen and J. N. Kok. *The Gaston Tool for Frequent Subgraph Mining.* In GraBaTs, pp: 77–87, 2004.

[28] C. R. Palmer, P. B. Gibbons and C. Faloutsos. *ANF: A Fast and Scalable Tool for Data Mining in Massive Graphs.* In SIGKDD, pp: 81-90, 2002.

[29] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu. *PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-projected Pattern Growth.* In ICDE, pp:215-224, 2001.

[30] M. P. H. Stumpf, C. Wiuf, and R. W. May. Subnets of scale free networks are not scale free: Sampling properties of networks. In PNAS, v 102, 2005.

[31] H. Toivonen. *Sampling Large Databases for Association Rules.* Proceedings of the 22th International Conference on Very Large Data Bases. pp: 134-145, 1996.

[32] C. Wang, W. Wang, J. Pei, Y. Zhu and B. Shi. *Scalable Mining of Large Disk-based Graph Databases.* In ACM SIGKDD, pp: 316-325, 2004.

[33] W. Wang, C. Wang, Y. Zhu, B. Shi, J. Pei, X. Yan and J. Han. *GraphMiner: A Structural Pattern-Mining System for Large Disk-based Graph Databases and Its Applications.* In ACM SIGMOD on management of data, pp: 879-881, 2005.

[34] T. Washio and H. Motoda. *State of the Art of Graph- Based Data Mining.* In ACM SIGKDD Explorations Newsletter, 5(1) pp: 59–68, 2003.

[35] X. Yan and J. Han. *gSpan: Graph-Based Substructure Pattern Mining*, In ICDM, pp: 721–724, 2002.

[36] M. Zaki, S. Parthasarathy, W. Li and M. Ogihara. *Evaluation of Sampling for Data Mining of Association Rules*. In RIDE, page 42, 1997.

[37] M. Zaki and C. J. Hsiao. CHARM: *An Efficient Algorithm for Closed Itemset Mining.* In SIAM, pp: 457-473, 2002.

[38] M. J. Zaki. Efficiently Mining Frequent Trees in a Forest. In ACM SIGKDD, pp: 71-80, 2002.