Sampling Massive Streaming Call Graphs

Shazia Tabassum
LIAAD, Inesc tec
University of Porto
Porto, Portugal
shazia.tabassum@inesctec.pt

João Gama
LIAAD, Inesc tec
University of Porto
Porto, Portugal
jgama@fep.up.pt

ABSTRACT

The problem of analyzing massive graph streams in real time is growing along with the size of streams. Sampling techniques have been used to analyze these streams in real time. However, it is difficult to answer questions like, which structures are well preserved by the sampling techniques over the evolution of streams? Which sampling techniques yield proper estimates for directed and weighted graphs? Which techniques have least time complexity etc? In this work, we have answered the above questions by comparing and analyzing the evolutionary samples of such graph streams. We have evaluated sequential sampling techniques by comparing the structural metrics from their samples. We have also presented a biased version of reservoir sampling, which shows better comparative results in our scenario.

We have carried out rigorous experiments over a massive stream of 3 hundred million calls made by 11 million anonymous subscribers over 31 days. We evaluated node based and edge based methods of sampling. We have compared the samples generated by using sequential algorithms like, space saving algorithm for finding topK items, reservoir sampling, and a biased version of reservoir sampling. Our overall results and observations show that edge based samples perform well in our scenario. We have also compared the distribution of degrees and biases of evolutionary samples.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning—*Concept learning*

Keywords

Data Streams, Streaming Graphs, Sampling, Call Graphs

1. INTRODUCTION

As described in [3] a data stream is an ordered sequence of instances that can be read only once or a small number of times using limited computing and storage capabilities. These sources of data are characterized by being open-ended, flowing at high-speed, and generated by non stationary distributions in dynamic environments. In terms of such graph streams, sampling is the process of selecting a subset of streaming graph to represent the characteristics of the entire graph stream at a given point of time. Because of the time and space limitations, it is difficult to analyze and mine massive social streams in real time. Wherefore, samples are generated, so as to gain approximate solutions for real time queries. A number of algorithms have been introduced in this research area to sample streams [1], [7], [6], [10], [11] etc. The authors of corresponding research papers have evaluated their work in terms of time and space complexity, but not based on the structural properties of streams.

All the sampling techniques developed so far only preserve some of the properties of original streams. However, we do not know which properties of streams are preserved by which sampling methods, that would be appropriate for a specific application scenario. In [5] the authors compared static snapshots of samples generated by algorithms like random node and edge sampling, random walks and forest fire with aggregated datasets of about five hundred thousand edges. Their goal was to find a general sampling method that would match full set of properties of original graph so that sample can be used for simulations and experiments. Our goal in this work, is to find appropriate sample techniques for specific properties of graphs. The main focus of this work is applying sequential sampling algorithms to the streaming graphs scenario and analyze the samples over the evolution of stream. Consequently the samples can be used for real time queries and experiments. We have also presented a biased version of reservoir sampling algorithm which shows better comparative results.

In other works of [9], the authors investigated to prove that the subnets of scale free networks are not scale free, by random sampling of static network. [2] discussed sampling algorithms for pure topology types. [4] studied three types of methods for sampling Internet graphs with a reduction of 70 % of original graphs. However the previous works did not answer questions like, which sequential techniques preserve the structure of evolutionary streams? which sampling techniques best preserve the community structures for weighted graphs? Which are most time efficient? And which sampling methods are appropriate for generating approximate solutions for specific real time queries? Which sampling techniques preserve the relevant structural properties of graph specific for an application? Nevertheless, no pre-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SAC 2016, April 04-08, 2016, Pisa, Italy

© 2016 ACM. ISBN 978-1-4503-3739-7/16/04…$15.00

DOI: http://dx.doi.org/10.1145/2851613.2851654

vious works focused on the evolution analysis of samples and the graphs with weighted and bi-directional edges. In this work, we answer the above questions regarding samples of real time streams like call graphs of Telecommunication Networks (TN). We do this by plotting the measures used for comparing the structures and properties preserved by different samples generated in real time. We also deal with the weighted and Bi-directional properties of graph. Furthermore, we analyze the evolution of metrics over 31 days of Call graph streams. Enhancing the diversity of algorithms for sampling, we introduce an algorithm, by modifying the existing Random Sampling Algorithm using reservoir by [10].

2. METHODOLOGY

In order to find the best suitable sampling algorithms for a scale free network like telecommunication call graphs, we analyzed the evolution of stream by generating snapshots of 31 samples at the end of each day, each from the beginning. For example, first sample for a stream of 1 day, second sample for the total stream of day 1 and day 2 etc.

Identifying community structures has its real time applications like customer profiling, segmentation, targeted marketing, fraud detection etc for telecoms service providers. To answer queries like above we need to answer the questions like, which samples preserve community structure? In this work we have exploited the evolution of community structures for 31 samples by evaluating centrality measures such Average Degree Centrality (ADC) and Average Weighted Degree Centrality (AWDC) of graphs; In-degree Centrality (IC) and Out-degree Centrality (OC) centrality of nodes. Number of Communities in the sample graphs. Degree Distribution of the samples. We have also analyzed the network connectedness using metrics: Number of Connected Components (NC) and Average Component Size (ACS).

Then, we compare the time for computation by each of the techniques. Using the approx. time taken for one single pass over the stream of minimum size 7,845,201 and a maximum size of 15,440,707 calls streams. However we also present the results to process a single edge, for edge based methods.

For answering queries like, which samples are appropriate for finding top remunerative customers? Who are the top frequent callers? We use the generated samples to run real time queries for specific applications, like finding top influential players using Eigen Vector Centrality (EVC).

3. SAMPLING ALGORITHMS AND METHODS

In this section we present the algorithms that we have used for generating samples. These algorithms can be implemented by using two types of methods. One is node based sampling and the other is edge based sampling method. By using these two methods we also present results and observations to find an appropriate technique for generating real-time samples.

3.1 Sampling Algorithms

3.1.1 Space Saving Algorithm

The Space Saving Algorithm [6] is the most approximate and efficient algorithm for finding top frequent elements from the stream. The algorithm maintains partial interest of information as it monitors only a subset of elements from the stream. It maintains counters for every element in the sample and increments its count when the element re-occurs in the stream. If a new element is encountered in the stream it is replaced with an element with the least counter value and its count is incremented.

3.1.2 Reservoir Sampling

This is a well known algorithm of Reservoir Sampling (RS), denoted as Algorithm R in [10]. The author mentioned in this work that all the algorithms using a reservoir of elements from the original data to generate samples are a kind of reservoir sampling. In algorithm R the author maintained a reservoir of elements with a predefined sample size. In the streaming scenario, initially the reservoir is filled with the initial elements from the stream. Every element after that, is computed for the probability of being inserted and a random number is generated to pick an element already in the sample. If the probability of the new element is greater than the probability of the selected element then the new element replaces and old one, if not it is discarded. By the end every element in the sample is selected with equal probability. Consequently, the items are inserted into the reservoir with decreasing probability. Therefore, it leads to samples with very old items from the stream, as also discussed in [1].

3.1.3 Biased Random Sampling

We have known the random sampling techniques with all the elements in the sample with equal probability. In this section we present a biased random sampling technique where we sample items/objects with unequal probability. Biased Random Sampling (BRS) is based on the idea of reservoir sampling but it ensures that every item in the stream definitely enters the reservoir.

input : Unbounded *stream*
output: Realtime *sample* of size k

Filling the reservoir with first k items/objects;
for $i = 1$ **to** k **do**
  $sample[i] \leftarrow stream[i]$;
end
Inserting all new items into the stream;
while *stream!= EOF* **do**
  $i = i + 1$;
  $pos \leftarrow Random(1, k)$;
  $sample[pos] \leftarrow stream[i]$;
end

Algorithm 1: Biased Random Sampling Algorithm

Algorithm 1 represents BRS. As a general initial step, the reservoir is filled with the first items from the stream. Then, we do not compute the probability of later items, as every item definitely enters stream. For replacing an item already in the sample, a random number is generated between 0 and the size of reservoir. The element at the position of random number is replaced with the item from stream. Here the probability of every item entering the reservoir is equal as every item enters the stream, but the probability of every item in the reservoir is not equal. Hence, this technique is biased towards new items from the stream. It can be used in the scenarios where old items are considered stale or not useful. In [1] the author also presented a biased sampling

algorithm, where he flips a coin for every point arrived in the stream. In the event of success, the new point replaces an old point already in the sample at random, if not a new slot is created and the point is added to it. We did not assay this technique as we compared samples with fixed size.

3.2 Node Based Methods

Node based methods in general, sample a set of nodes from the original graph. The resultant samples contain a set of vertices from the graph stream and showing no connections between them. The samples posses only nodes and no structure. To evaluate this method, we have implemented the space saving algorithm by sampling top frequent nodes and call it as (SSN), detailed in section 5.

3.3 Edge Based Methods

As the name suggest, these samples are generated by selecting a subset of edges from the original graph. The resultant graph is a subgraph of original graph with nodes and edges. The algorithms that can be implemented in the nodes based methods, can also be implemented using edge based methods in our scenario. We have conducted rigorous experiments implementing algorithms in section 3.1 using edge selection for sampling, i.e. RS, BRS and space saving algorithm by sampling edges (SSE). In section 5, we discussed the experiments in detail.

4. CASE STUDY

4.1 Telecommunication Networks

Sampling on graphs have been studied in different domains, with petty work in telecommunication networks. In this work, we apply the sampling algorithms in this scenario. Call Detail Records (CDR's) from TN are one of the largest and fastest data streams with stupendous mass of information hidden. We made use of such anonymous CDR stream of 386,492,749 calls made by 11,916,442 subscribers over 31 days . Spread across 24 hrs per day and gathered from geographically distributed sources.

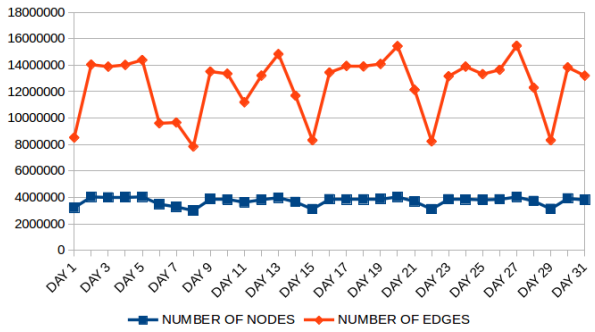

Figure 1: Evolution of nodes and edges in our graph stream

4.2 Semantics of Call Graphs

We modeled telecommunication call graphs as nodes corresponding to callers and callees. The edges between them represent calls. These edges can be weighted using frequency of calls, duration of call, etc. In our scenario, we have weighted our samples based on frequency of calls. The edges are bidirectional, corresponding to incoming and outgoing calls. The snapshots of number of nodes and edges per day stream are shown in figure 1. The figure shows decreased call activity on Sundays compared to other days of week.

5. EXPERIMENTAL EVALUATION

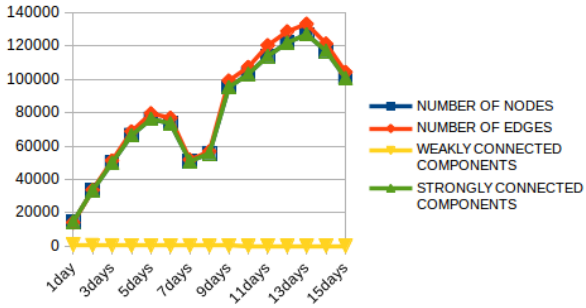

Figure 2: Evolution of nodes and edges using SSN

To evaluate the node based methods we have used the SSN by sampling top frequent callers. We generated 15 sample snapshots for 15 days of stream each from the beginning day 1. The samples generated using this method have only nodes and no structure, therefore we also acquire the corresponding edges of nodes in the real time. As the number of edges incident upon the sampled nodes increases, so are the adjacent nodes. As a result, we have a subgraph with increased number of nodes, derived from the associated edges. The time for computation of such methods also increases substantially with the added time for acquiring edges and their corresponding nodes. The evolution of number of nodes, edges and properties of graph are represented in figures 2 and 3. We did not proceed with the other algorithms for node based method, because of the space and time complexity mentioned above.

For evaluating edge based methods we have generated 31 sample snapshots of size 10^4 edges from 31 days of call graph streams each from the beginning. This is done for three sampling algorithms using edge selection, i.e SSE, RS and BRS. Number of nodes and edges in the resultant subgraphs are shown in figures 4 and 5. Figure 5 shows number of distinct edges in each sample. BRS samples contains less number

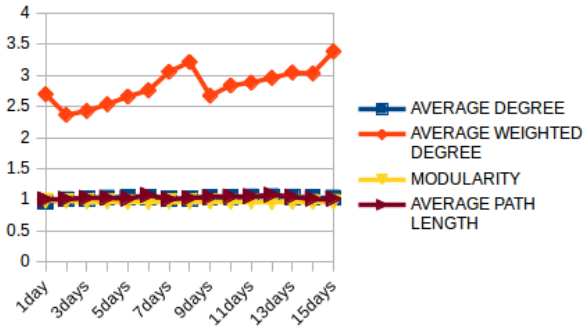

Figure 3: Evolution of metrics using SSN

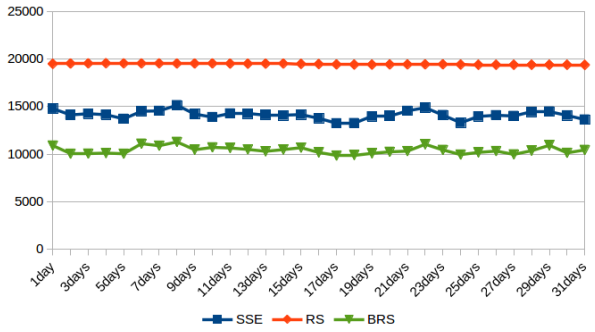

Figure 4: Number of nodes

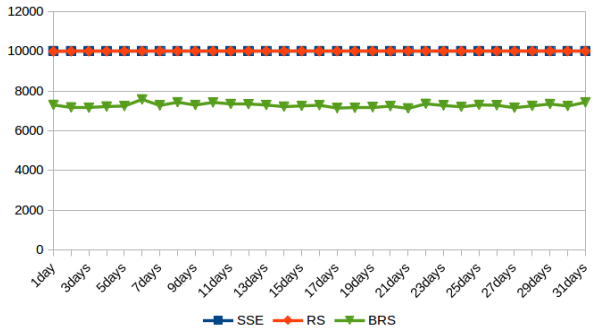

Figure 5: Number of edges

of distinct edges and more number of repetitive edges/calls exhibiting calling behavior of callers. This indicates BRS samples are good at generating weighted samples based on frequency. In our scenario, we map the multi-graph of calls onto weighted network, thus the weighted edges indicate frequency of calls between two nodes/callers. SSE samples unweighted edges, as it selects the top frequent edges but not the frequency of edges. Subgraphs of RS has a negligible amount of weighted edges based on frequency of calls.

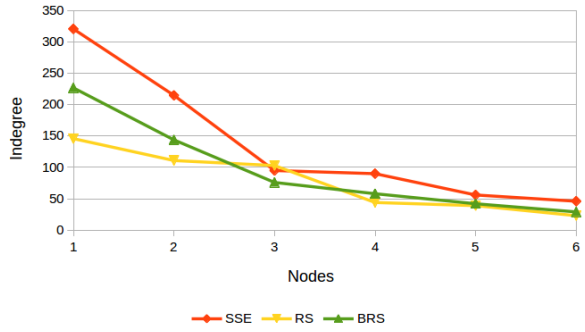

Figure 6: In-degree Centralities

Figure 6 and 7 plots the IC and OC of top 6 IC and OC nodes of three samples, we observe that SSE samples contain nodes with high IC's and OC's while RS samples have nodes with least IC and OC. When we compare both the IC and OC we find that all the three samples are biased towards high in-degree nodes and low out-degree nodes. This suggest

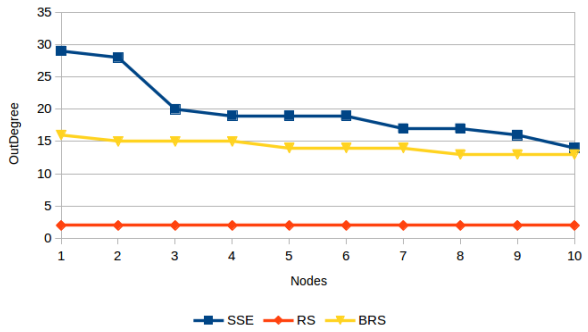

Figure 7: Out-degree Centralities

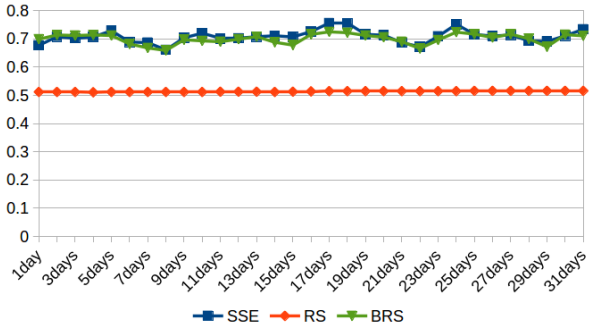

Figure 8: Average Degree Centrality

the structure of original graph with high number of incoming calls and less number of outgoing calls for each node on an average.

5.1 Community Structure

In our experiments we have used degree centralities to evaluate the community structures in subgraphs. Figure 8 depicts the ADC of the three methods. RS shows least average degree centrality than SSE and BRS which are almost the same. This suggests that RS is biased towards very low degree nodes. While SSE and BRS show similar degree centralities throughout the evolutionary stream. Figure 9 plots the AWDC of three samples. Comparing both the above referenced plots, we can observe that the ADC's of SSE and RS are similar to their respective AWDC's. While the AWDC of BRS is more compared to its ADC. When ADC and AWDC are similar, the edges in the samples have no weights. When AWDC is greater than ADC the samples contain weighted edges. In both the figures 8 and 9, SSE and BRS show low graph centralities on Sundays as related to the original graph with low activity on Sundays. While RS curve shows no deviations.

Figure 10 plots the logarithmically binned degree distribution of nodes in the samples. We observe that 99.4% of nodes from the RS sample has degree 1. which indicates that large number of nodes are sparsely connected displaying least community structure. Figure 11, depicts the number of communities in each sample, from which we notice that RS contains maximum number of communities. From both the above results, we infer that RS samples have maximum number of communities with minimum degree than SSE and BRS. The above results confer that RS has least community

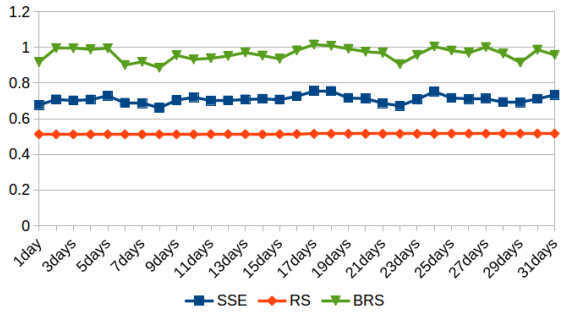

Figure 9: Average Weighted Degree Centrality

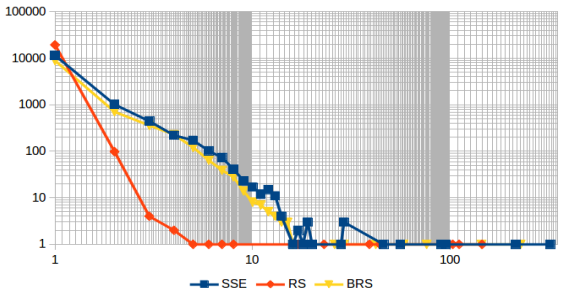

Figure 10: Degree Distribution

structure with more number of nodes sparsely connected.

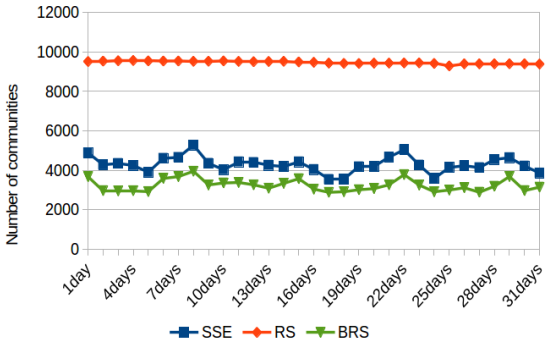

Figure 11: Number of Communities

5.2 Component Structure

Figure 12 shows NC's in the three samples. We identify RS samples contain maximum NC's for all the days. BRS sample contain least NC's. Figure 13 plots the ACS of each sample, where SSE and BRS has similar ACS samples and greater compared to RS. From both the figures it is evident that RS samples exhibit least component structure with more NC's having least ACS compared to SSE and RS.

5.3 Time Complexity

To measure the time complexity of three algorithms we have used two metrics, one is min time and other is max time. Minimum time for one pass over the stream of 7845201 edges and maximum time for one pass over the stream of

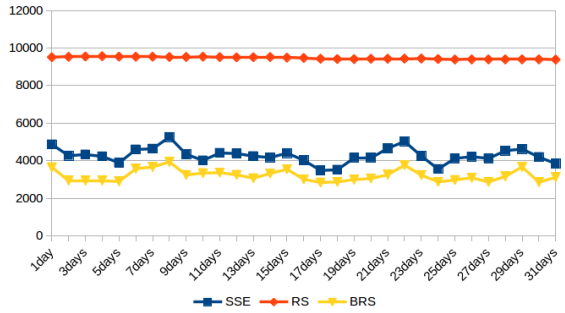

Figure 12: Number of Components

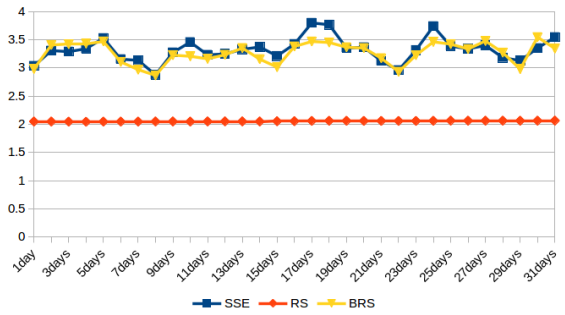

Figure 13: Average Component Size

15468336 edges. Figure 14 plots the time for computation by three algorithms using same hardware, from which it is apparent that the time for computation by SSE is maximum. RS takes the least time for computation while BRS with slightly more. However the time for processing each edge as it arrives in real-time is much lower to approx. 15 ms to 20 ms for SSE, 0.5 to 1 ms for BRS and 0.3 ms to 0.4 ms for RS.

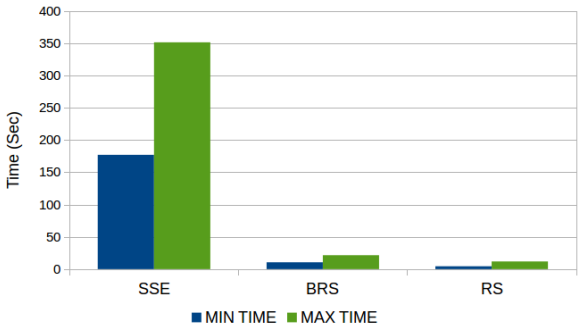

Figure 14: Time Complexity

5.4 Running Real Time Queries

Which samples are best suitable for running real-time queries and getting approximate answers to queries like, who are the top influential players in the network? To answer questions like these we have used EVC to compute top influential nodes from the three samples generated by SSE, RS and BRS. All the three samples generate similar and accurate results for top 6 nodes, evident from figure 15.

As analyzed by [8] call graphs exhibit a power-law distribution with few nodes displaying high activity and majority of nodes displaying least activity. The above results imply that the referenced sampling techniques capture highest activity nodes. However, the results of the three samples begin to differ as the number of top influential nodes increases.

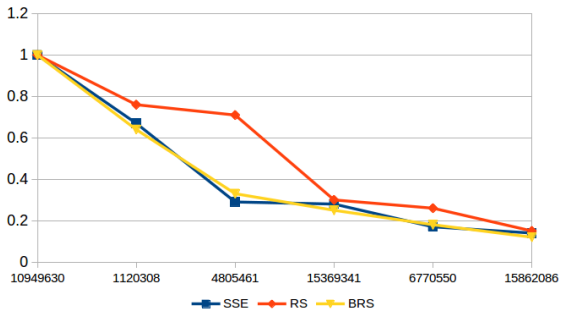

Figure 15: Eigen Vector Centrality

6. EMPIRICAL OBSERVATIONS

After analyzing the samples generated using four techniques discussed in section 3, we observe that: For generating structured subgraphs from node based methods, we need to acquire the edges associated with nodes, which increases the time for computation. The evolution analysis of temporal samples generated using SSN shows a gradual increase in the size of samples. For edge based methods, we observe that RS is biased to nodes with low degree centralities and BRS and SSE nodes exhibit higher degree centralities compared to it. BRS best suits for measuring weighted centralities based on frequency of edges. Hence, it is also suitable for running real-time queries for finding frequent items over the sample. BRS and SSE sample communities with high average degree centralities. That shows a better community structure when compared to RS. Therefore BRS and SSE would be more suitable for applications analyzing community structure. SSE and BRS generates samples with better component structure compared to RS. RS and BRS has good performance with runtime compared to SSE. For using samples to run queries like top frequent items, SSE would be appropriate as it samples top frequent edges, while not considering other factors. All the three sampling algorithms give similar results for applications like finding most minimum number of top influential nodes using EVC.

7. CONCLUSION

Our purpose in this work, was not to find the best sample, instead we have compared the samples to find which is an appropriate sampling technique for a specific application scenario. We analyzed the properties of samples to compare the sampling techniques. We have used a real-time massive stream to evaluate the evolutionary samples. So, that real time queries and experiments can run over those samples. Different sampling algorithms have been evaluated in this work, based on structures, properties, time complexity and applications. We have evaluated weighted graph samples. We have also proposed an algorithm with the modification over Random Sampling Algorithm using Reservoir. We have found that on an average of all the measures our proposed algorithm has exhibited better performance. Our results drive many observations related to the biases and appropriateness of sampling techniques. We have also analyzed the evolution of samples that can be used for evolution prediction in future works.

Acknowledgements

Authors acknowledge the support of the European Commission through the project MAESTRA (Grant Number ICT-750 2013-612944) and FCT - Portuguese Foundation for Science and Technology within project UID/EEA/50014/2013. The authors also thank WeDo Business for providing the data.

8. REFERENCES

[1] C. C. Aggarwal. On biased reservoir sampling in the presence of stream evolution. In *Proceedings of the 32nd international conference on Very large data bases*, pages 607–618. VLDB Endowment, 2006.

[2] E. M. Airoldi and K. M. Carley. Sampling algorithms for pure network topologies: a study on the stability and the separability of metric embeddings. *ACM SIGKDD Explorations Newsletter*, 7(2):13–22, 2005.

[3] J. Gama. *Knowledge discovery from data streams*. CRC Press, 2010.

[4] V. Krishnamurthy, M. Faloutsos, M. Chrobak, L. Lao, J.-H. Cui, and A. G. Percus. Reducing large internet topologies for faster simulations. In *NETWORKING 2005. Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Systems*, pages 328–341. Springer, 2005.

[5] J. Leskovec and C. Faloutsos. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 631–636. ACM, 2006.

[6] A. Metwally, D. Agrawal, and A. El Abbadi. Efficient computation of frequent and top-k elements in data streams. In *Database Theory-ICDT 2005*, pages 398–412. Springer, 2005.

[7] M. Papagelis, G. Das, and N. Koudas. Sampling online social networks. *Knowledge and Data Engineering, IEEE Transactions on*, 25(3):662–676, 2013.

[8] R. Sarmento, M. Oliveira, M. Cordeiro, J. Gama, and S. Tabassum. Social network analysis of streaming call graphs. In *Big Data Analysis: New Algorithms for a New Society*, page In Press. Springer, 2015.

[9] M. P. Stumpf, C. Wiuf, and R. M. May. Subnets of scale-free networks are not scale-free: sampling properties of networks. *Proceedings of the National Academy of Sciences of the United States of America*, 102(12):4221–4224, 2005.

[10] J. S. Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985.

[11] W. Wei, J. Erenrich, and B. Selman. Towards efficient sampling: Exploiting random walk strategies. In *AAAI*, volume 4, pages 670–676, 2004.