Sampling from Large Graphs

Jure Leskovec
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA, USA
jure@cs.cmu.edu

Christos Faloutsos
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA, USA
christos@cs.cmu.edu

ABSTRACT

Given a huge real graph, how can we derive a representative sample? There are many known algorithms to compute interesting measures (shortest paths, centrality, betweenness, etc.), but several of them become impractical for large graphs. Thus graph sampling is essential.

The natural questions to ask are (a) which sampling method to use, (b) how small can the sample size be, and (c) how to scale up the measurements of the sample (e.g., the diameter), to get estimates for the large graph. The deeper, underlying question is subtle: how do we measure success?

We answer the above questions, and test our answers by thorough experiments on several, diverse datasets, spanning thousands nodes and edges. We consider several sampling methods, propose novel methods to check the goodness of sampling, and develop a set of scaling laws that describe relations between the properties of the original and the sample.

In addition to the theoretical contributions, the practical conclusions from our work are: Sampling strategies based on edge selection do not perform well; simple uniform random node selection performs surprisingly well. Overall, best performing methods are the ones based on random-walks and "forest fire"; they match very accurately both static as well as evolutionary graph patterns, with sample sizes down to about 15% of the original graph.

Categories and Subject Descriptors: H2.84 [Database Management]: Database Applications – Data Mining

General Terms: Measurement, Experimentation

Keywords: graph sampling, graph mining, scaling laws

1. INTRODUCTION

Given a large massive graph with millions or billions of nodes, how can we create a small, but "good" sample out of it? In many applications we need to run expensive algorithms, like simulations of internet routing protocols, peer-to-peer gossip-like protocols, virus propagation and immunization policies, or analysis of "viral marketing" scenarios. For example, in studies of Internet routing protocols computer communication researchers would like to do detailed simulations of BGP (Border Gateway Protocol), or flow level simulations, but the simulations on networks with more than a few thousand nodes may be prohibitively expensive [4].

The questions we ask here are:

- What is a good sampling method? Should we pick random nodes? Random edges? Use some other strategy?
- What is a good sample size?
- How do we measure the goodness of a single sample, as well as the goodness of a whole sampling method?

For the last question, we need to address two aspects: What do we compare against? Do we want the sample graph S to have similar (or scaled-down) properties as compared to the original graph G? Or, do we want the sample S to be similar to what the graph G looked like *back in the time* when it had the size of S? We refer to the former goal as Scale-down goal, and to the latter as Back-in-time goal.

Next issue is to define a list of graph properties that we should aim for. The properties would be, say, the shape of the degree distribution should be heavy tailed; small diameter, etc. Our goal here is not to find a sampling procedure and the corresponding (unbiased) estimator (scaling rule) for a *single* property of the graph (e.g. number of edges, diameter). We are more interested in finding a *general sampling method* that would match a *full* set of graph properties so that sampled graphs can be used to for simulations and more complicated/expensive experiments.

The main results of this work are the following:

- We propose two different goals on sampling: the Back-in-time goal and the Scale-down goal.
- For each of the above goals, we provide the most thorough analysis and comparison in the published literature, testing multiple sampling algorithms (10), on several real datasets (5), with 14 graph properties and patterns, using 2 different evaluation methods.
- We perform a systematic evaluation of sampling algorithms, introducing non-trivial statistical evaluation methods (the Kolmogorov-Smirnov D-statistic and random walk inspired ideas), that go beyond simple eye-balling.

Best performing sampling methods are the following: for the Scale-down sampling goal, methods based on random walks perform best, since they are are biased towards high degree nodes and give sampled graphs that are connected. For the Back-in-time sampling goal, we find out that "Forest Fire" type sampling and sampling based on PageRank score of a node perform best. These methods are not as biased and thus nicely mimic the temporal evolution of the graph.

KDD'06, August 20–23, 2006, Philadelphia, Pennsylvania, USA.
Copyright 2006 ACM 1-59593-339-5/06/0008 ...$5.00.

2. RELATED WORK

Sampling on graphs has been used in many different flavors but very little has been done on matching a large set of graph properties. Previous work focused on using sampling to condense the graph to allow for better visualization [7, 12]. Works on graph compression focused on transforming the graph to speed up algorithms [6]. Techniques for efficiently storing and retrieving the web-graph were also studied [1]. Internet modeling community [8] studied sampling from undirected graphs and concluded that some graph properties can be preserved by random-node selection with sample sizes down to 30%. A recent work [2] studies separability and stability of various graph properties for a number of different graph generation algorithms.

A seemingly related, but vastly different problem comes from the area of web-crawling or P2P networks, where the question is, how to select a random node from a graph, if we only see a neighborhood [15]. This problem is not related to ours, because we do have the full graph, and we can easily pick nodes at random – the challenge for us is to select a set of nodes so that the induced graph obeys general characteristics, and so the sampled graphs can be used for simulations and further analysis.

3. PROPOSED METHOD

3.1 Problem definition

In graph sampling we are given a large directed *target* graph and the task is to create a small *sample* graph, that will be similar (have similar properties). There are two ways to look at the graph sampling: under the *Scale-down* goal we want to match the static target graph, while under the *Back-in-time* goal we want to match its temporal evolution.

3.1.1 Scale-down sampling goal

In *Scale-down* sampling we are given a large static *directed* graph G on n nodes. We are also given the size of the sample n'. The goal is to create a sample graph S on n' nodes, $n' \ll n$, that will be most similar to G, i.e. we want S to have similar graph properties as G. For example, similar degree distribution and/or diameter. We precisely define "similar" in section 3.2.

3.1.2 Back-in-time sampling goal

The *Back-in-time* sampling goal corresponds to traveling back in time and trying to mimic the past versions of G, of which we only get to observe the final, static snap-shot. Note that we also do not know the ages of nodes and edges. Let $G_{n'}$ denote graph G at some point in time, when it had exactly n' nodes. Now, we want to find a sample S on n' nodes that is most similar to graph $G_{n'}$, i.e. when graph G was of the same size as S.

The hard part here is that we want to match patterns describing the temporal evolution together with the patterns defined on a single snapshot of a graph, which also change over time. If one would have node ages, then the best possible approach would be to simply roll-back the evolution (addition/deletion of nodes and edges over time). Note that our sampling algorithms *do not know* the age of individual nodes and edges. So the question here is whether we can roll-back the time without having any temporal information (age of nodes/edges).

3.2 Evaluation techniques

3.2.1 Criteria for a static snapshot of a graph

First, we present a set of static graph patterns, which are measured on a single snapshot of a graph. Given a graph, we measure the following nine graph properties. Essentially we treat all as *distributions* to allow for proper scaling:

- S1: In-degree distribution: for every degree d, we count the number of nodes with in-degree d. Typically it follows a power-law and some other heavy tailed distribution [5].
- S2: Out-degree distribution.
- S3: The distribution of sizes of weakly connected components ("wcc"): a set of nodes is weakly connected if for any pair of nodes u and v there exists an *undirected* path from u to v.
- S4: The distribution of sizes of strongly connected components ("scc"): a set of nodes is strongly connected, if for any pair of nodes u and v, there exists a *directed* path from u to v and from v to u.
- S5: Hop-plot: the number $P(h)$ of reachable pairs of nodes at distance h or less; h is the number of hops [11].
- S6: Hop-plot on the largest WCC.
- S7: The distribution of the first left singular vector of the graph adjacency matrix versus the rank.
- S8: The distribution of singular values of the graph adjacency matrix versus the rank. Spectral properties of graphs often follow a heavy-tailed distribution [3].
- S9: The distribution of the clustering coefficient C_d [16] defined as follows. Let node v have k neighbors; then at most $k(k-1)/2$ edges can exist between them. Let C_v denote the fraction of these allowable edges that actually exist. Then C_d is defined as the average C_v over all nodes v of degree d.

3.2.2 Criteria for temporal graph evolution

We also study the five temporal graph patterns, that are measured on a *sequence* of graphs over time. Essentially, we measure a single number (e.g., diameter) of a graph and see how it evolves with the graph size. Thus, we treat all these criteria as distributions: each is a set of numbers, with one number for each time-tick (a desirable sample size n').

- T1: Densification Power Law (DPL) [9]: number of edges versus the number of nodes over time. DPL states that $e(t) \propto n(t)^a$. The *densification exponent* a is typically greater than 1, implying that the average degree of a node in the graph is *increasing* over time.
- T2: The effective diameter of the graph over time, which is defined as the minimum number of hops in which 90% of all connected pairs of nodes can reach each other. It has been observed that the effective diameter generally shrinks or stabilizes as the graph grows with time [9].
- T3: The normalized size of the largest connected component (CC) over time.
- T4: The largest singular value of graph adjacency matrix over time.
- T5: Average clustering coefficient C over time [16]: C at time t is the average C_v of all nodes v in graph at time t.

3.2.3 Statistical tests for graph patterns

To compare the two graph patterns (static or temporal) we use the Kolmogorov-Smirnov D-statistic. Usually D-statistics is applied as a part of Komogorov-Smirnov test to reject the null hypothesis. Here we simply use it to measure the agreement between the two distributions. It is defined as $D = \max_x\{|F'(x) - F(x)|\}$, where x is over the range of the random variable, and F and F' are the two empirical *cumulative distribution functions* of the data. Note that the D-statistic does not address the issue of the scaling but rather compares the shape of the (normalized) distribution.

Evaluation procedure: we use the D-statistic for both the Scale-down and Back-in-time sampling goals to measure the agreement between the true and the sample graph property:

For the Scale-down sampling goal, we are given a sample graph S on n' nodes and a target G on n nodes, $n' \ll n$. We measure the 9 distributions for static graphs, on both G and S, and compare them using the D-statistic. We logarithmically bin the x-axis for all the distributions S1-S9.

In case of Back-in-time sampling goal, we are given a sequence G_t of snap-shots of a temporally evolving graph G. We are also given a sequence of sample graphs S_t, where for every t, S_t and G_t have the same number of nodes. For each t, we measure 9 distributions for static graphs on G_t and S_t and compare them. We also measure 5 temporal graph patterns on both sequences of graphs and compare them using the D-statistic. Given that we have data on graphs over time, we can perform the exact evaluation.

3.2.4 Visiting probability

The intuition we use here is that in a good sample the probability that a random walk, which starts at node v, visits node w should be similar in sample S and target G. For every node v in G (S) we calculate the the stationary probability of a random walk starting at node v. We then use Frobenius norm to measure the difference in visiting probability. Due to space limitations we refer the reader to extended version of the paper for details on these experiments.

3.3 Sampling algorithms

Conceptually we can split the sampling algorithms into three groups: methods based on randomly selecting nodes, randomly selecting edges and the exploration techniques that simulate random walks or virus propagation to find a representative sample of the nodes.

As it will turn out in section 4, for Scale-down sampling goal random walks perform best, while in Back-in-time sampling "Forest Fire" performs best.

3.3.1 Sampling by random node selection

First, we introduce three sampling algorithms based on random node selection. The most obvious way to create a sample graph is to uniformly at random select a set of nodes N and a sample is then a graph induced by the set of nodes N. We call this algorithm the *Random Node (RN)* sampling. Authors in [14] showed that RN does not retain power-law degree distribution. Note that here we are not interested in limiting behavior or discussions whether a distribution is a power-law or not. We simply want to find best sampling method that gives the most similar degree distribution. So, our conclusions are valid even if the original graph deviates from exact power-law (which is often the case in practice).

In contrast to uniform we also explore non-uniform sampling strategies. One way is to set the probability of a node being selected into the sample to be proportional to its *PageRank weight*. We refer to this as *Random PageRank Node (RPN)* sampling. *Random Degree Node (RDN)* sampling has even more bias towards high degree nodes. Here the probability of a node being selected is proportional to its degree. Intuitively this method will have problems matching degree distribution, since there are too many high degree nodes in the sample. We also expect it to give samples that are very dense.

3.3.2 Sampling by random edge selection

Similarly to selecting nodes at random, one can also select edges uniformly at random. We refer to this algorithm as *Random Edge (RE)* sampling. There are several problems with this idea: sampled graphs will be very sparsely connected and will thus have large diameter and will not respect community structure. A slight variation of random nodes is *Random Node-Edge (RNE)* sampling, where we first uniformly at random pick a node and then uniformly at random pick an edge incident to the node. Intuitively, RE sampling is slightly biased to high degree nodes, since they have more edges incident to them. RNE does not have this bias.

Based on recommendation from [8] we also implemented the *Hybrid (HYB)* approach, where with probability p we perform a step of RNE sampling and with probability $1-p$ we perform a step of RE sampling. We set $p = 0.8$, which was found to perform best [8].

3.3.3 Sampling by exploration

The common idea in this family of sampling techniques is that we first select a node uniformly at random and then explore the nodes in its the vicinity.

In *Random Node Neighbor (RNN)* sampling we select a node uniformly at random together with all of its out-going neighbors. This is suitable for very large disk resilient graphs since it imitates reading an edge file. As we will see it matches well the out-degree distribution, but fails in matching in-degrees and the community structure.

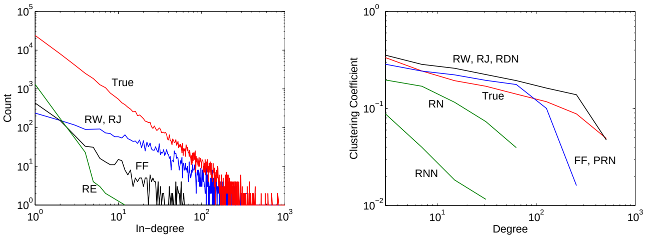
Next, we present two ideas based on random walks. First, in *Random Walk (RW)* sampling, we uniformly at random pick a starting node and then simulate a random walk on the graph. At every step with probability $c = 0.15$ (the value commonly used in literature) we fly back to the *starting* node and re-start the random walk. There is problem of getting stuck, for example, if the starting node is a sink, and/or it belongs to a small, isolated component. The solution is: If, after a very long number of steps, we do not visit enough nodes to meet the required sample size, we select another starting node and repeat the procedure. In our experiments we run the random walk for $100 * n$ steps.

A very similar idea to Random Walk sampling is the *Random Jump (RJ)* sampling. The only difference here is that with probability $c = 0.15$ we randomly jump to *any* node in the graph. Note that this algorithm does not have problems of getting stuck or not being able to visit enough nodes.

The *Forest Fire (FF)* sampling is inspired by the work on temporal graph evolution. We randomly pick a *seed* node and begin "burning" outgoing links and the corresponding nodes. If a link gets burned, the node at the other endpoint gets a chance to burn its own links, and so on recursively. Model has two parameters: forward (p_f) and backward (p_b) burning probability. The exact definition is given in [9].

3.3.4 Other sampling strategies

One could easily think of other simple sampling strategies. In particular, authors in [8] also explore contraction based methods, and depth and breath first search graph traversal, but none of them performed well overall.



(a) S1: In-degree (b) S9: Clustering coef.

Figure 1: Scale-down sampling goal

One could also use ideas from graph clustering and graph partitioning. These algorithms are usually computationally intensive and do not scale well to very large graphs. Graph sampling becomes important with massive graphs, where real-world algorithms become too expensive and one has to reside to sampling. Thus simplicity of sampling is essential.

4. EXPERIMENTAL EVALUATION

In the following section we present the results of experiments on several real graphs. We show results for both Scale-down and Back-in-time sampling goals. We also explore how the quality of the sample degrades with the sample size and conclude with a set of scaling rules that help is scale the properties of the sample to the target graph.

4.1 Dataset description

We consider five different datasets: a static graph and four graph with temporal information, which allows us to also evaluate them for the Back-in-time sampling goal.

Citation networks: The HEP–TH and HEP–PH citation graphs from the e-print arXiv and covers all the citations within a dataset of n=29,555 papers with e= 352,807 citations, and n=30,567, e=348,721 respectively. The data in both graphs covers papers from Jan 1992 to Apr 2003 (10 years) and represents essentially the complete history of the section. For each year Y ($1992 \leq Y \leq 2003$) we create a citation graph using all papers published before year Y.

Autonomous systems: A graph of routers comprising the Internet can be organized into sub-graphs called Autonomous Systems (AS) [10] . The dataset contains 735 daily instances spanning 785 days from November 8 1997 to January 2 2000. Graphs range in size from n=3,011 and e=10,687 to n=6,474 and e=26,467. To mimic the temporal evolution we took 20 graphs uniformly spanning the number of nodes between the largest and the smallest AS graph.

Bipartite affiliation network: Using the arXiv data, we also constructed a bipartite *affiliation graph*. There is a node for each paper and for each author, and an edge connecting people to the papers they authored. The network is derived from the astro physics category in the arXiv. We place a time-stamp on each node: the submission date of each paper, and for each person, the date of their first submission to the arXiv. The data for affiliation graph covers the period from April 1992 to March 2002. It has n=57,381 nodes (19,393 authors, 37,988 papers) and e=133,170 edges.

Network of trust is a static snapshot of a who-trusts-whom network from `epinions.com` [13]. We only had access to a single snapshot of the network, which contains n=75,879 nodes and e=508,960 edges.

4.2 Scale-down sampling goal

We begin by introducing the experimental results on Scale-down sampling, where the goal is to create a sample graph S, that will match the properties of the target graph G.

4.2.1 Matching the graph patterns

First we present illustrative examples of behavior of sampling algorithms from section 3.3. We run each algorithm ten times on each of five datasets and plot the patterns.

Figure 1(a) shows the in-degree distribution of target and the sampled graphs for a sample size of 10%. For presentational purposes we do not plot the results of all 10 sampling algorithms. We notice three qualitatively different behaviors: RDN, RJ and RW sampling algorithms observe very similar behavior – we only show RW. These algorithms are biased towards high degree nodes and densely connected parts of the graph. Observe how the tail of degree distribution is over estimated at the cost of under represented nodes with small degrees. The second class consists of FF, RPN and RN (we only plot FF), which are not biased towards high degree nodes. Observe that degree distribution is well matched and has about the same shape (slope) as the true degree distribution. The last cluster are RE, RNE and HYB, which are all based on random edge selection. When the size of the sample is small, the resulting samples are very sparsely connected, the high degree nodes are under-represented and the slope of degree distribution is too steep. We notice roughly the same three classes of behavior with the same members for most of our experiments.

Figure 1(b) plots the logarithmically binned distribution of clustering coefficient. The plot shows the average fraction of triangles present around the nodes of degree d. Intuitively, it gives us clues about the community structure in the graph. We make the following observations:

The sample size is small, so the RN and RNN are far off. Edge selection techniques (RE, RNE and HYB) give too sparse samples which contain no triangles. FF works fine.

4.2.2 Evaluation using the D-statistic

Next, we evaluate the sampling algorithms against all 14 graph static and temporal patterns. We present the results in table 1. Each entry in the table is obtained by averaging the D-statistic over 10 runs, 5 datasets, and 20 sample sizes per dataset. For Scale-down sampling criteria the temporal patterns are essentially flat, since regardless of a sample size we want to match the property of the final graph.

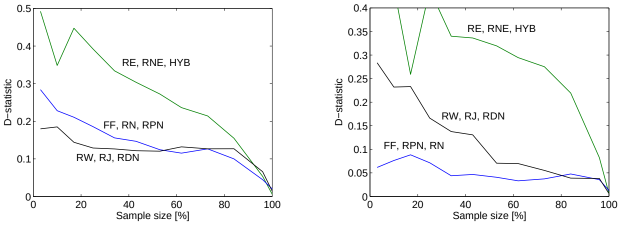
For each column we bolded the best scoring algorithm. Notice that FF, RW and RPN fit well the degree distribution. Distribution of weakly connected components is best matched by edge selection techniques. The distribution of the number of reachable pairs of nodes at particular distance (column denoted as *hops*) is best matched by FF and RPN. Singular values and first left singular vector of graph adjacency matrix are best approximated using RW and FF. Exploration methods match the clustering coefficient.

For temporal graph patterns in table 1 we see that RW performs best. This means that properties of sample graphs obtained by RW algorithm, have very stable and flat temporal patterns, e.g. as we see on figure 2 the diameter remains almost constant and the graph is highly connected.

We omit the results for difference in random walk node visiting probability between the sample and the target graph. We note the results were very similar to those on table 1.

	Static graph patterns								Temporal graph patterns				
	in-deg	out-deg	wcc	scc	hops	sng-val	sng-vec	clust	diam	cc-sz	sng-val	clust	**AVG**
RN	0.084	0.145	0.814	0.193	0.231	0.079	0.112	0.327	0.074	0.570	0.263	0.371	0.272
RPN	**0.062**	**0.097**	0.792	0.194	**0.200**	0.048	0.081	0.243	0.051	0.475	0.162	0.249	0.221
RDN	0.110	0.128	0.818	0.193	0.238	0.041	0.048	0.256	0.052	0.440	**0.097**	0.242	0.222
RE	0.216	0.305	**0.367**	0.206	0.509	0.169	0.192	0.525	0.164	0.659	0.355	0.729	0.366
RNE	0.277	0.404	0.390	0.224	0.702	0.255	0.273	0.709	0.370	0.771	0.215	0.733	0.444
HYB	0.273	0.394	0.386	0.224	0.683	0.240	0.251	0.670	0.331	0.748	0.256	0.765	0.435
RNN	0.179	0.014	0.581	0.206	0.252	0.060	0.255	0.398	0.058	0.463	0.200	0.433	0.258
RJ	0.132	0.151	0.771	0.215	0.264	0.076	0.143	**0.235**	0.122	0.492	0.161	**0.214**	0.248
RW	0.082	0.131	0.685	0.194	0.243	0.049	**0.033**	**0.243**	**0.036**	**0.423**	**0.086**	0.224	**0.202**
FF	0.082	0.105	0.664	0.194	**0.203**	**0.038**	0.092	**0.244**	0.053	0.434	0.140	**0.211**	**0.205**

Table 1: Scale-down sampling criteria. On average RW and FF perform best.



(a) Scale-down (b) Back-in-time

Figure 3: Scale-down sampling goal

We set the RW random walk returning probability to high value of 0.8, which means we were basically performing breath first search from the starting node. For FF we set $p_f = 0.7$ and $p_b = 0$, so the fire burned 2.3 nodes on average.

All in all the results suggest that there is no single perfect answer to graph sampling. Depending on particular application and properties of interest one would choose appropriate method, with exploration methods performing best overall.

4.3 Back-in-time sampling goal

A second approach to sampling is the Back-in-time sampling goal. Here we do not to match the properties of the final target graph G, but rather match G as it grew and evolved. This means that we compare the patterns from a sample graph S on n' nodes with the target graph G, when it was of the same size as S.

4.3.1 *Matching the graph properties*

Figure 2(a) plots the Densification Power Law. Again notice 3 types of behavior. RW and RDN give too dense graphs, edge selection techniques (RE, RNE and HYB) result in too sparse graphs that start to rapidly fill-in the missing edges when the sample size gets large. FF, RN and RPN match the temporal densification of the true graph.

Plot for the effective diameter on figure 2(b) shows that edge selection techniques give samples with too high diameter. On contrary, RW, RJ and RDN have constant diameter, which is good for Scale-down sampling. And, FF, RN and RPN match the shrinking diameter of the true graph over time quite well. Similar observations can be done for the size of connected component and average clustering coefficient over time (figures 2(c), (d)). RW, RJ and RDN give connected graph with constant clustering coefficient. Edge selection techniques produce disconnected graphs and thus underestimate the clustering. FF, RN and RPN match the size of connected component and clustering over time.

4.3.2 *Evaluation using the D-statistic*

We use the same evaluation strategy as in previous section. We refer to extended version of the paper for the exact results. Overall, FF performed best (average D-statistic 0.13), closely followed RPN(0.14). Second group was then formed by RN, RW with D-stat of 0.16. Again, edge selection performed worst. We obtained best results when setting FF $p_f = 0.2$, which means it burned 0.25 nodes on average.

4.3.3 *Sensitivity analysis of parameter values*

For RW and RJ the probability of random jump was set to 0.15. For Forest Fire (FF) and Back-in-time goal, we obtain good results for $0 \leq p_f \leq 0.4$, where D-statistic is below 0.16, obtaining the best D-statistic of 0.13 at $p_f = 0.20$. For Scale-down goal best performance was obtained with high values of p_f ($p_f \geq 0.6$) where every fire eventually floods the connected component.

4.4 How small sample to take?

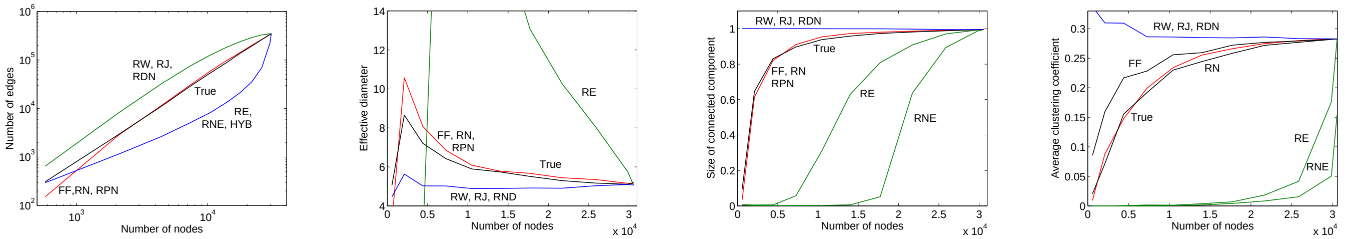
Last, we explore how the quality of the sample degrades with the sample size. Figure 3 shows performance of sampling algorithms for both Scale-down and Back-in-time sampling goals. We plot the D-statistic as a function of sample size. Notice that edge selection techniques perform bad on both Scale-down and Back-in-time sampling goals. For Scale-down sampling, up to 50% sample size exploration and node selection techniques perform about the same. As the sample size gets smaller, RW, RJ and RDN start to outperform FF, RN and RPN. For Back-in-time sampling goal the situation is reversed. Performance of RW, RJ and RDN sampling algorithms slowly degrades as the sample size decreases. On the contrary, FF, RN and RPN perform much better. Also notice that quality of the sample decreases much slower. For Back-in-time sampling goal we are able to obtain good samples of size only around 15 percent.

4.5 Scaling rules

Assuming we use Random Walk (RW) sampling for Scale-down and Forest Fire (FF) sampling for Back-in-time goal, with moderate sampling sizes (≈ 25%), we find the scaling rules shown in table 2. Most of them are very simple, since the sampling does a good job of matching the patterns.

5. CONCLUSION

Generating smaller, "realistic" samples of larger graphs is an important tool for large graph analysis, and what-if scenarios. Besides the task of coming up with a good sampling algorithm our work addresses a subtle questions: What do



(a) T1: DPL (b) T2: Diameter vs. time (c) T3: Size of CC vs. time (d) T5: Clustering vs. time

Figure 2: Back-in-time sampling goal for HEP–PH citation graph. FF and RPN match well the true patterns.

Static patterns	Scale-down	Back-in-time
S1, S2: Degree distr.		matches target G
S5, S6: Hop-plot	scale with sample size	obeys evolution
S7: Singular vector	same for top components	obeys evolution
S8: Singular values	scale with sample size	scale with sample size
S9: Clustering coef.	matches target	obeys evolution
Temporal patterns		
T1: DPL		obeys evolution
T2: Diameter	constant	obeys evolution
T3: Fraction of CC	constant	obeys evolution
T5: Avg. clustering	constant	obeys evolution

Table 2: Empirical rules for scaling graph patterns

we mean by “realistic”? How do we measure the deviation from realism? There seems to be no perfect single answer to graph sampling. Depending on particular application and properties one cares about a suitable algorithm is chosen.

The contributions of this work are the following:

• We propose the Back-in-time goal for sampling which is better defined than the Scale-down sampling goal.

• We study a large variety of graph sampling algorithms. Methods that do a combination of random node selection, and a little of vicinity exploration give best samples. “Forest Fire” is best performing sampling algorithm. Best results are obtained with burning probability ≈0.7 for Scale-down, and ≈0.2 for Back-in-time. This means that for Scale-down larger fires exploring more of the node’s vicinity help, while for Back-in-time smaller fires which explore less vicinity give better results.

• We show that a 15% sample is usually enough, to match the properties (S1–S9 and T1–T5) of the real graph.

• We recognize that no list of properties will ever be perfect. We give a long list of “realism criteria”, inspired from our experience with social networks, citation networks and communication networks. Our list is a significantly larger superset of the lists of earlier work on graph sampling.

• We provide rules to scale-up the measurements from the sample, to reach the measurements of the original graph.

Future work could focus on sampling from graphs with weighted edges, as opposed to the current, zero-one weights, as well as sampling on graphs with labeled edges. A different direction would be to focus on sampling algorithms to estimate a single property of the graph (e.g., diameter, number of triads) and define precise estimators (scaling relations) and evaluate their variance and bias.

ACKNOWLEDGEMENTS

We thank Michalis Faloutsos of UCR for useful discussions. Work supported by the NSF grants IIS-0209107, SENSOR-0329549, EF-0331657, IIS-0326322, IIS-0534205. by the Pennsylvania Infrastructure Technology Alliance, and a gift from Hewlett Packard. Jure Leskovec was partially supported by a Microsoft Research Graduate Fellowship.

6. REFERENCES

[1] M. Adler and M. Mitzenmacher. Towards compressing web graphs. In *Data Compression Conference*, 2001.

[2] E. M. Airoldi and K. M. Carley. Sampling algorithms for pure network topologies. *SIGKDD Explor.*, 2005.

[3] D. Chakrabarti, Y. Zhan, and C. Faloutsos. R-mat: A recursive model for graph mining. In *SDM*, 2004.

[4] X. A. Dimitropoulos and G. F. Riley. Creating realistic BGP models. *IEEE/ACM MASCOTS*, 2003.

[5] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *SIGCOMM*, pages 251–262, 1999.

[6] T. Feder and R. Motwani. Clique partitions, graph compression and speeding-up algorithms. In *Journal of Computer And System Sciences*, volume 51, 1995.

[7] A. C. Gilbert and K. Levchenko. Compressing network graphs. In *LinkKDD*, 2004.

[8] 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.

[9] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: Densification laws, shrinking diamaters and possible explanations. In *ACM SIGKDD*, 2005.

[10] U. of Oregon. Route views project.

[11] C. R. Palmer, P. B. Gibbons, and C. Faloutsos. Anf: A fast and scalable tool for data mining in massive graphs. In *SIGKDD*, Edmonton, AB, Canada, 2002.

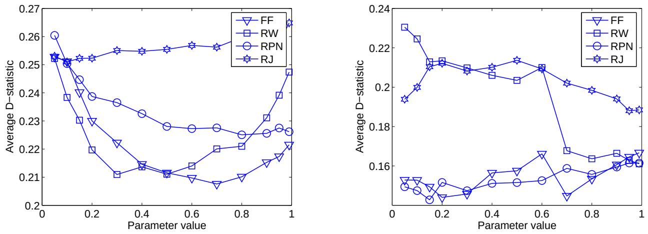
[12] D. Rafiei and S. Curial. Effectively visualizing large networks through sampling. In *Visualization*, 2005.

[13] M. Richardson, R. Agrawal, and P. Domingos. Trust management for the semantic web. In *Second International Semantic Web Conference*, 2003.

[14] M. P. H. Stumpf, C. Wiuf, and R. M. May. Subnets of scale-free networks are not scale-free: Sampling properties of networks. In *PNAS*, volume 102, 2005.

[15] D. Stutzbach, R. Rejaie, N. Duffield, S. Sen, and W. Willinger. Sampling techniques for large, dynamics graphs. In *CIS-TR-06-01, University of Oregon*, 2006.

[16] D. J. Watts and S. H. Strogatz. Collective dynamics of ’small-world’ networks. *Nature*, 393:440–442, 1998.



(a) Scale-down

(b) Back-in-time

Figure 4: Parameter sensitivity for Scale-down and Back-in-time sampling goals.

APPENDIX

A. APPENDIX

Here we present additional material on graph sampling that is not presented in the conference version of the paper.

A.1 Forest Fire sampling algorithm

First, we give the exact definition and the algorithm for the Forest Fire sampling.

We first choose node v uniformly at random. We then generate a random number x that is geometrically distributed with mean $p_f/(1-p_f)$. Node v selects x out-links incident to nodes that were not yet visited. Let $w_1, w_2, \ldots, w_x$ denote the other ends of these selected links. We then apply this step recursively to each of $w_1, w_2, \ldots, w_x$ until enough nodes have been burned. As the process continues, nodes cannot be visited a second time, preventing the construction from cycling. If the fire dies, then we restart it, i.e. select new node v uniformly at random. We call the parameter p_f the forward burning probability.

A.2 Back-in-time sampling goal

Table 3 shows the Kolmogorov-Smirnov D-statistic for Back-in-time sampling. For each of the four temporally evolving graphs we run each of ten sampling algorithms ten times and report average D-statistic over all runs and sample sizes. This means each cell in table 3 is the average over more than 500 different runs of sampling.

Notice how Forest Fire (FF) sampling algorithm performs best for the most of the graph patterns. Also, notice that even for static patterns it performs better than node selection and exploration methods in Scale-down sampling goal (table 1). This means that static patterns also exhibit evolution, e.g. degree distribution changes as the graph grows, and Forest Fire is able to capture this evolutionary pattern.

For Back-in-time sampling we obtained best results when setting FF $p_f = 0.2$. The Forest Fire burned 0.25 nodes on the average, which means it explored very little of the node's vicinity and was pretty much similar to random node sampling.

A.3 Parameter sensitivity

Figure 4 shows the average Kolmogorov-Smirnov D-statistic for Scale-down and Back-in-time sampling goals. We vary the parameter value: for Forest Fire this is the forward burning probability (p_f), for Page Rank Random Node, Random Walk and Random Jump sampling this is the probability of a random surfer jumping to a random node or back to starting node in case of Random Walk sampling.

Notice that for Forest Fire and the Scale-down goal (figure 4(a)) best performance was obtained with high values the burning probability: $p_f \geq 0.6$. Random Jump performs poorly all the time. Random Walk works best for low parameter values.

In case of Back-in-time sampling goal (figure 4(a)) we observe the following: For Forest Fire (FF) and Back-in-time goal, we obtain good results for $0 \leq p_f \leq 0.4$, where D-statistic is below 0.16, obtaining the best D-statistic of 0.13 at $p_f = 0.20$. Notice that for both the Page Rank Random Node sampling and the Forest Fire the quality of sampling gets worse as one increases the parameter value (burning / teleportation probability). For Random Jump and Random Walk sampling we obtain best results with high jumping probability, which means the sampling algorithm behaves very much like sampling nodes uniformly at random.

	Static graph patterns							Temporal graph patterns					
	in-deg	out-deg	wcc	hops	sng-val	sng-vec	clust	DPL	diam	cc-sz	sng-val	clust	**AVG**
RN	0.032	0.094	0.469	0.196	0.062	0.077	0.310	0.063	0.075	0.088	0.116	**0.290**	0.156
RPN	**0.021**	0.072	0.453	0.124	0.048	0.063	0.291	**0.023**	0.052	0.056	0.104	**0.303**	**0.134**
RDN	0.118	0.148	0.460	**0.280**	0.054	0.064	0.318	0.091	0.081	**0.045**	0.127	0.391	0.181
RE	0.179	0.275	0.636	0.555	0.147	0.156	0.509	0.225	0.221	0.223	0.239	0.598	0.330
RNE	0.244	0.371	0.650	0.740	0.224	0.211	0.661	0.215	0.380	0.430	0.101	0.579	0.400
HYB	0.242	0.362	0.649	0.719	0.205	0.193	0.630	0.246	0.357	0.374	0.169	0.644	0.399
RNN	0.138	0.088	0.535	0.213	**0.040**	0.206	0.399	0.116	**0.047**	**0.045**	**0.077**	0.329	0.186
RJ	0.130	0.157	**0.309**	0.252	0.090	0.136	0.303	0.086	0.168	0.141	0.122	0.382	0.190
RW	0.090	0.116	0.340	0.219	0.044	0.076	0.304	0.082	0.054	0.066	0.128	0.430	0.162
FF	**0.020**	**0.070**	0.447	**0.125**	0.049	**0.058**	**0.281**	**0.022**	0.058	0.055	0.099	0.361	**0.137**

Table 3: Back-in-time sampling criteria. *D*-statistic for 10 sampling algorithms (section 3.3) average over 4 temporally evolving graphs. Bolded the best performing sampling algorithms based on *D*-statistic. Column AVG shows the average over all patterns. FF and RPN perform best followed by RNand RW. Edge selection performs poorly.