Sampling from Large Graphs

~ Paper Review ~

Adam Cristian - MOC 2

Motivation

- There are many known algorithms to compute interesting measures in graph theory, but several of them become impractical when are applied to large networks.
- Therefore, how it is possible to derive a representative sample?
- To answer this question, the authors needed to find answers first to the next questions:
  - What is a good sampling method? Should it pick random nodes? Random edges? Use some other strategy?
  - What is a good sample size?
  - How the precision of a single sample is measured, as well as the precision of a sampling method?

Article Goal

- The paper goal is not to find a sampling procedure and the corresponding (unbiased) estimator (scaling rule) for a single property of the graph.
- The authors are more interested in finding a general sampling method that would match a full set of graph properties.

- Another questions pop up: how the sampled graphs are compared against? The sample graph S should have similar (or scaled-down) properties as compared to the original graph G? Or the sample S should be similar to what the graph G looked like back in the time when it had the size of S?
- The article refers to the former goal as a Scale-down goal and the latter as a Back-in-time goal.

Problem Definition

- A large directed target graph is given and the task is to create a small sample graph, that will be similar (have similar properties).
- In Scale-down sampling, a large directed graph G having n nodes is given. The size of the sample n' is also known. The goal is to create a sample graph S on n' nodes, that will be most similar to G and has almost the same properties as G.
- The Back-in-time sampling goal corresponds to travelling back in time and trying to mimic the past versions of G and only the final version of the original graph is known.
- The hard part here is to match patterns describing the temporal evolution of G together with the patterns defined on a single snapshot of the graph, which also changes over time. Therefore, the question here is whether the time can be rolled back without having any temporal information (age of nodes/edges).

Sampling Algorithms

- The sampling algorithms can be split conceptually into three groups:
  - methods based on randomly selecting nodes
  - approaches based on randomly selecting edges
  - the exploration techniques that simulate random walks or virus propagation to find a representative sample of the nodes

Sampling by 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. This algorithm is called the Random Node (RN) sampling. This method does not retain power-law degree distribution.
- In contrast to uniform, another approach for selection is 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. This method is called Random PageRank Node (RPN) sampling.
- Random Degree Node (RDN) sampling has even more bias towards high degree nodes. Intuitively this method will have problems matching degree distribution in the cases where there are many high degree nodes in the sample.

Sampling by Random Edge Selection

- Similarly to selecting nodes at random, one can also select edges uniformly at random. This algorithm is referred to as Random Edge (RE) sampling. There are several problems with this idea: sampled graphs will be very sparsely connected and will thus have a large diameter and will not respect community structure.
- A slight variation of random nodes is Random Node-Edge (RNE) sampling, where at first is picked uniformly at random a node and then uniformly at random picked an edge incident to the node.
- The authors also implemented the Hybrid (HYB) approach, where with probability p a step of RNE sampling is performed and with probability 1 − p a step of RE sampling is done.

Sampling by Exploration

- The main idea is that at first a node is selected uniformly at random and then explore the nodes in its vicinity.
- In Random Node Neighbor (RNN) sampling a node is selected uniformly at random together with all of its out-going neighbours. It is easy to notice that this algorithm matches well the out-degree distribution but fails in matching in-degrees and the community structure.
- In Random Walk (RW) sampling, the method uniformly at random picks a starting node and then simulates a random walk on the graph. At every step with probability c = 0.15 (the value commonly used in literature), the process flies back to the starting node and re-start the random walk.
- 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 the method randomly jump to any node in the graph. 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 done on temporal graph evolution. This approach randomly picks 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. The model has two parameters: forward and backward burning probability.

Evaluation Techniques

- The next issue is to define a list of graph properties that the task aims for.
- Set of static graph patterns, which are measured on a single snapshot of a graph:
  - S1: In-degree distribution for every degree
  - S2: Out-degree distribution
  - S3: The distribution of sizes of weakly connected components
  - S4: The distribution of sizes of strongly connected components
  - S5: Hop-plot: the number P(h) of reachable pairs of nodes at distance h or less
  - S6: Hop-plot on the largest weakly connected component
  - 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
  - S9: The distribution of the clustering coefficient C
- Five temporal graph patterns, that are measured on a sequence of graphs over time:
  - T1: Densification Power Law (DPL): number of edges versus the number of nodes over time
  - T2: The effective diameter of the graph over time
  - T3: The normalized size of the largest connected component over time
  - T4: The largest singular value of graph adjacency matrix over time
  - T5: Average clustering coefficient C over time

Statistical tests for graph patterns

- The D-statistic metric is used to compare the two graph patterns (static or temporal). Usually, D-statistics is applied as a part of the Kolmogorov-Smirnov test to reject the null hypothesis. In the paper, the authors simply use it only to measure the agreement between the two distributions.
- For the Scale-down sampling goal, all the 9 distributions are measured for static graphs and the original one and compared them using the D-statistic.
- In the case of the Back-in-time sampling goal, all 9 distributions are measured for static graphs and the original one at that time period and compared. The authors also measure 5 temporal graph patterns on both sequences of graphs and compare them using the D-statistic.

Dataset

- The paper tests the answers to the initial questions by thorough experiments on several, diverse datasets, spanning thousands of nodes and edges.
- Five different datasets are considered: a static graph and four graphs with temporal information, which allows the evaluation for the Back-in-time sampling goal.
  - Citation networks: The HEP–TH and HEP–PH citation graphs from the e-print arXiv
  - Autonomous systems
  - Bipartite affiliation network based on arXiv dataset
  - Network of trust (the only graph without temporal data)

Results - Scale-down Sampling Criteria

- It's noteworthy that FF, RW and RPN fit well the degree distribution. The distribution of weakly connected components is best matched by edge selection techniques. The distribution of the number of reachable pairs of nodes at a particular distance (column denoted as hops) is best matched by FF and RPN. Singular values and the first left singular vector of the graph adjacency matrix are best approximated using RW and FF. Exploration methods match the clustering coefficient.

	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**

Results - Back-in-time Sampling Criteria

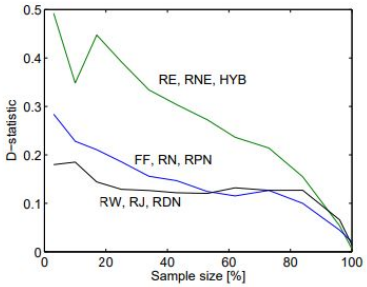
- Overall, Forest Fire performed best (average D-statistic 0.13), closely followed Random PageRank Node(0.14). The second group was then formed by Random Node, Random Walk with a D-statistic value of 0.16. Again, edge selection performed worst. The authors obtained the best results when setting Forest Fire forward burning probability to 0.2, which means it burned 0.25 nodes on average.

	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**

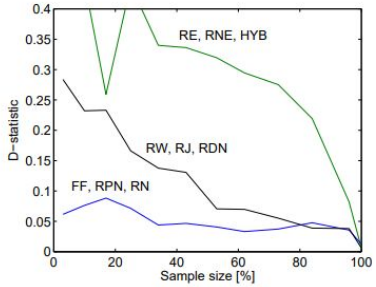
Computational Results

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

Sample Size



(a) Scale-down



(b) Back-in-time

- The article also explores how the quality of the sample degrades with the sample size. The figure shows the performance of sampling algorithms for both Scale-down and Back-in-time sampling goals.
- Notice that edge selection techniques perform badly 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 the Back-in-time sampling goal, the situation is reversed. Also, it is noticeable that the quality of the sample decreases much slower. For the Back-in-time sampling goal, the authors are able to obtain good samples of size only around 15 per cent.

Conclusions

- Graph sampling becomes important with massive graphs, where real-world algorithms become too expensive and one has to reside to sampling. Therefore simplicity of sampling is essential.
- Generating smaller, “realistic” samples of larger graphs is an important tool for large graph analysis, and what-if scenarios.
- There seems to be no perfect single answer to graph sampling. Depending on particular applications and the graph’s properties of interest or that are the most relevant ones, a suitable algorithm is chosen.