Representative Subgraph Sampling using Markov Chain Monte Carlo Methods

Christian Hübler[1], **Karsten Borgwardt**[2], **Hans-Peter Kriegel**[1], **Zoubin Ghahramani**[2]
[1] Ludwig-Maximilians-Universität München, [2] University of Cambridge

Abstract

Bioinformatics and the Internet keep generating graph data with thousands of nodes. Most traditional graph algorithms for data analysis are too slow for analysing these large graphs. One way to work around this problem is to sample a smaller 'representative subgraph' from the original large graph.

Existing representative subgraph sampling algorithms either randomly select sets of nodes or edges, or they explore the vicinity of a randomly drawn node. All these existing approaches do not make use of topological properties of the original graph and provide good samples down to sample sizes of approximately 15% of the number of nodes in the original graph. In this article, we propose novel sampling methods for representative subgraph sampling, based on the Metropolis algorithm and Simulated Annealing. The key idea is to find a subgraph that preserves properties of the original graph that are efficient to compute or to approximate.

In our experiments, we improve over the pioneering work of Leskovec and Faloutsos (KDD 2006), by producing representative subgraph samples that are both smaller and of higher quality than those produced by other methods from the literature.

1. Introduction

In many application domains it is desirable to study a representative subgraph rather than the entire original graph, because the original graph with thousands of nodes might be too large for traditional graph algorithms that are often designed for graphs with tens of nodes. We will present algorithms for this task of 'representative subgraph sampling', with 'representative' describing the requirement that the sample shall preserve crucial graph properties of the original graph. This problem can be cast into the following optimization problem:

$$\text{argmin}_{|G'|=n'} \Delta(S(G'), S(G))$$

where G is the original graph, G' the subgraph sample, $S(X)$ is a topological property of graph X, and Δ is a distance function on these topological properties. n' is the desired size of the subgraph sample that has to be prespecified.

Preliminary work. Under review by the International Workshop on Mining and Learning with Graphs (MLG). Do not distribute.

2. Metropolis Graph Sampling

From a statistical point of view, *induced* graph sampling is the task to draw a set x of n' nodes from the $n := |V(G)|$ nodes of the original graph G. For example in the case of *RandomNode* (Leskovec & Faloutsos, 2006) the samples x are uniformly distributed on the sample space $\mathfrak{X} = \{x \subset V(G) | n' = |x|\}$.

The main idea of Metropolis graph sampling is to draw a sample from the sample space $\mathfrak{X}$ following a specific density $\varrho(x)$. This density should reflect subgraph sample quality well, which means good induced samples $G' = G(x)$ should be drawn more frequently than worse ones. Thus $\varrho(x)$ depends on the quality of the sample $G(x)$. From this point of view an obvious choice of $\varrho(x)$ is one depending on a distance measure with respect to a preprocessed graph property. The severe problem with such a density $\varrho(x)$ is that it is only given in an unnormalized manner $\varrho^*(x)$. To obtain the normalized density $\varrho(x)$ we have to calculate and sum over $\binom{n}{n'}$ summands which is obviously intractable:

$$\varrho(x_1) = \frac{\varrho^*(x_1)}{\sum_{x \in \mathfrak{X}} \varrho^*(x)}$$

How can we draw samples from the sample space $\mathfrak{X}$ if the underlying normalized density $\varrho(x)$ is not given explicitly? This problem is solved by the Metropolis algorithm (Metropolis et al., 1953). In general Metropolis can draw samples from any probability distribution, provided that a function $f(x)$ proportional to the density $\varrho(x)$ (in our case inversely proportional) can be calculated for any

Algorithm 1 Metropolis Subgraph Sampling

Input: Graph G, distance function $f(\cdot) = \Delta_G(\cdot)$, sample size n', number of possible transitions $\#it$, exponent p
  $x \leftarrow$ uniformly at random from $\mathfrak{X}$
  $best_x \leftarrow x$
  $best_{f(x)} \leftarrow f(x)$
  for $i := 1$ to $\#it$ **do**
    $y \leftarrow$ uniformly at random from adjacent states $A(x)$
    $\alpha \leftarrow$ uniformly at random from interval $[0, 1]$
    if $\alpha < (\frac{f(x)}{f(y)})^p$ **then**
      $x \leftarrow y$
      if $f(x) < best_{f(x)}$ **then**
        $best_x \leftarrow x$
        $best_{f(x)} \leftarrow f(x)$
      end if
    end if
  end for
Output: $G(best_x)$

$x \in \mathfrak{X}$. In our case $f(x)$ can be any distance measure between topological properties of the sample $G(x)$ and the original graph G, for instance, a distance on degree distributions.

We use the Metropolis algorithm for optimization rather than for approximating a distribution. We turn Metropolis into an optimization algorithm by choosing an appropriate unnormalized density $\varrho^*(x)$, which in our case is inversely proportional to a distance measure $f(x) = \Delta(G, G(x)) = \Delta_G(x)$ between the real graph G and the sample $G(x)$. As the search space is exponential in the size of the subgraph sample, we have to reward 'good' samples extremely because otherwise lower-quality samples would dominate the process of sampling due to their large number. We do this by exponentiating the difference $f(x)$ by a large positive scalar p. Hence we define ϱ^* as

$$\varrho_p^*(x) := \frac{1}{f(x)^p} = \frac{1}{\Delta_G(x)^p} = \frac{1}{\Delta(G, G(x))^p}, \quad (1)$$

where $p \in \mathbb{R}^+$ and $p \gg 0$ (see pseudocode in Algorithm 1).

How is a new state y generated in our Metropolis Subgraph Sampling procedure? We delete one node from the current subgraph sample and all its adjacent edges. Then we randomly pick a new node from the graph and insert it (and its edges adjacent to the sample nodes) into our subgraph sample.

Speeding up convergence A common way to avoid that the Metropolis algorithm gets stuck in local optima is to use *Simulated Annealing* which gradually increases the exponent with proceeding runtime. Since Simulated Annealing avoids local optima which can slow down convergence, it may be useful for speeding up the sampling process. We define the density for Simulated Annealing as

$$\varrho_{p,T}^*(x) = e^{\frac{\log \varrho_p^*(x)}{T}} = e^{log f(x)^{(-\frac{p}{T})}} = f(x)^{(-\frac{p}{T})},$$

where T is the temperature parameter that is gradually decreased towards zero during Simulated Annealing.

We also propose a new graph-specific approach called *Chaining* to speed up convergence: As mostly the real graph G is connected or at least consists of few connected components, it can be observed that good samples are extremely often connected. Thus Chaining is restricting the search space $\mathfrak{X}$ to connected samples. While reducing the size of the search space, Chaining obviously requires additional checks whether a subgraph sample that is created by node deletion and insertion is still connected.

3. Experimental evaluation

3.1. Sampling

We test our described algorithms on 5 datasets representing graph models of a protein, the Internet, two social networks and a PPI-network. The 329 to 75879 nodes of these real-world graphs are connected with 2100 to 420899 edges. On these 5 datasets, we ran 15 different sampling strategies to generate a representative subgraph sample with $n' = 100$ nodes each.[1] These 15 approaches included 4 state-of-the-art methods for representative subgraph sampling: RandomNode (RN) and RandomEdge (RE), that randomly sample nodes and edges from the graph, and ForestFire, once for sampling induced subgraphs (FF_i), once for sampling non-induced subgraphs (FF).

We compared these methods to our novel approaches to representative subgraph sampling: Metropolis subgraph sampling (M), and Chaining (CH). We ran Metropolis once each for approximating the degree distribution (M_d), the graphlet distribution (M_g), the clustering coefficient (M_c) and for the unweighted combination (M_{dcg}) and a weighted combination (M_{10dcg}) of these three 3 criteria.

We set ϱ^* as in equation (1). For the two criteria which gave the best results using Metropolis, namely degree distribution ($_d$) and weighted combination ($_{10dcg}$), we repeated the same experiments using Chaining (Ch_d and Ch_{10dcg}). To assess the effect of Simulated Annealing both on runtime and sample quality, we performed Metropolis and Chaining using a Simulated Annealing Schedule, both for the degree distribution (M_d^{SA} and Ch_d^{SA}) and the weighted combination (M_{10dcg}^{SA} and Ch_{10dcg}^{SA}).

For our experiments, we use Simulated Annealing with a

[1] We have implemented all described algorithms in Java. All tests were performed on a P4 with 2.6 GHz and 2 GB main memory.

geometric annealing schedule, which means that we gradually reduce the temperature in the following manner:

$$T_{t+1} = \gamma T_t \text{ with } 0 < \gamma < 1$$

where t is the t-th step in our sampling procedure.

3.2. Parameter settings

Before we discuss our results, we provide the crucial parameter settings here that allow the reproduction of our findings.

Forest Fire The quality of samples obtained by *ForstFire* depend on the accurate choice of the forward burning probability p_f. In (Leskovec & Faloutsos, 2006) a forward burning probability $p_f > 0.6$ is proposed for sampling. This is consistent with our results in initial test runs, hence we use $p_f = 0.7$.

Metropolis The number of iterations $\#it$, which describes the maximum number of transitions performed by the Markov Chain, is set to 10,000 and the exponent p is determined via $p = 10 \cdot \frac{k}{n} \log_{10} n$, with k being the number of edges of G.

When using a combination of degree distribution, graphlet distribution and clustering coefficient, we consider an unweighted sum of the distances on them (M_{dcg}), or a weighted sum in which the distances on the degree distribution get 10 times more weight than those on graphlets and clustering coefficient (M_{10dcg}), as the degree distribution is the criterion that reaches the best results on its own. When using these weighted or unweighted combinations of three graph properties, we performed 20,000 iterations instead of 10,000 to be sure that the Markov Chain will converge and the pre-calculation of 3 graph properties was not in vain.

Chaining As Chaining remarkably speeds up convergence, but requires additional runtime to check connectedness of samples, we only perform one third of the iterations executed in standard Metropolis.

3.3. Evaluation

To measure the quality of the sampled subgraphs, we compute their distance to the original graph in terms of the degree distribution (degree), the diameter (diam), the clustering coefficient (clust), and the graphlet distribution (graphlet). In addition, we compute the mean of these 4 distances, which gives us the average distance, denoted by AVG. We report sample quality in terms of these distances between the subgraph sample and the original graph in Table 1 as averages over 25 repetitions on all 5 datasets.

In addition to the quality of the sample, the runtime t (in seconds) of each algorithm is shown in the last column. Because all proposed algorithms need to precompute some graph properties, the amount of time (also in seconds) this pre-calculation requires is stated as t_{prep}. t_{read} denotes the time required to read the original graph. t_{run} is the runtime of the actual sampling stage.

Sample Quality As can be seen from Table 1 (Column AVG), 8 out of the 11 variants of our Metropolis sampling algorithms outperform all existing state-of-the-art sampling algorithms in terms of sample quality. They preserve the graph properties of the original graph in a subgraph sample of size $n' = 100$ better than all other methods.

Metropolis It is a remarkable fact that M_d, although approximating the degree distribution of the original graph G only, is able to approximate the clustering coefficient vector and the graphlet distribution of G on average better than any of the state-of-the-art methods. Even in terms of the approximating the diameter, it is second best only to FF_i among the existing methods.

In terms of runtime, our methods are fast despite the need to compute graph properties: Even on the largest graph (Epinions, 75879 nodes), our slowest method (M_{10dcg}) generates a high-quality subgraph sample in ~ 367 seconds. Our fastest method M_d^{SA} generates a high-quality sample in 2 seconds on the same dataset. M_d is faster than the other variants of Metropolis (M_c, M_g, M_{dcg}, and M_{10dcg}), as the degree distribution can be computed more efficiently by parsing the adjacency list representation of a graph once.

Chaining Chaining leads to a speed-up in runtime if its computational overhead — guaranteeing connectedness of subgraph samples — is smaller than its computational savings by speeding up convergence. This speed-up is observable if the graph properties that we want to approximate and that have to be determined in each iteration of our sampling procedure are rather expensive to compute.

Simulated Annealing In our empirical evaluation, Simulated Annealing led to a slightly lower runtime on average, but also a slight loss in sample quality compared to basic Metropolis. We investigated why Simulated Annealing is empirically faster than Metropolis in our setting and noticed that in the first iterations of Simulated Annealing, low values of p lead to very sparse samples whose graph properties could be computed more efficiently than those of the graph samples generated by Metropolis.

Still, Simulated Annealing outperforms on average all state-of-the-art methods with respect to sample quality without the need to pick an exponent p.

4. Conclusions

In this article, we have proposed novel approaches to representative subgraph sampling. They are based on the key idea to compute or approximate properties of the original

	degree	diam	clust	graphlet	AVG	t_{read}	t_{prep}	t_{run}	t
RN	0.911	0.653	0.332	0.540	0.487	0.491	0.000	0.008	0.499
RE	0.725	0.633	0.389	0.500	0.449	0.491	0.000	0.009	0.500
FF	0.450	0.455	0.311	0.268	0.297	0.491	0.000	0.004	0.495
FF_i	0.340	0.157	0.132	0.280	0.227	0.491	0.000	0.012	0.503
M_d	**0.105**	0.247	0.121	0.143	0.154	0.491	0.001	3.096	3.587
M_c	0.814	0.498	0.072	0.287	0.418	0.491	5.227	2.857	8.575
M_g	0.788	0.531	0.405	0.177	0.475	0.491	64.572	4.430	69.493
M_{dcg}	0.544	0.326	0.069	0.072	0.253	0.491	69.237	35.088	104.816
M_{10dcg}	0.164	0.220	**0.063**	0.084	0.132	0.491	62.365	110.277	173.133
M_d^{SA}	0.147	0.252	0.121	0.156	0.169	0.491	0.001	2.613	3.104
M_{10dcg}^{SA}	0.214	0.256	0.067	0.072	0.147	0.491	69.173	95.761	165.425
Ch_d	0.182	0.172	0.119	0.077	0.137	0.491	0.001	8.965	9.456
Ch_{10dcg}	0.189	0.149	0.087	0.090	**0.129**	0.491	63.073	37.257	100.821
Ch_d^{SA}	0.204	**0.139**	0.139	0.086	0.142	0.491	0.000	8.830	9.321
Ch_{10dcg}^{SA}	0.221	0.181	0.111	**0.065**	0.148	0.491	62.993	32.923	96.407

Table 1. Distances between properties of subgraph sample and original graph ($n' = 100$). Each row is a sampling strategy, each column a graph property used for evaluation (left of the double bar). Numbers are distances between subgraph sample and original graph (Left columns). Runtimes for each method are given in seconds (Right columns). For full details on all abbreviations see Section 3.

graph G, which are then to be approximated well by the sample graph G'. While existing sampling algorithms manage to produce good samples with a minimum size of 15% (of nodes of the original graph) (Leskovec & Faloutsos, 2006), our algorithms succeed in constructing representative samples of smaller size (down to 0.14% on Epinions).

References

Leskovec, J., & Faloutsos, C. (2006). Sampling from large graphs. *KDD*.

Metropolis, N., Rosenbluth, A., Rosenbluth, N., & Teller, A. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics, Volume 21, Number 6* (pp. 1087–1092).