# Metropolis Algorithms for Representative Subgraph Sampling

Christian Hübler, Hans-Peter Kriegel
Institute of Computer Science
Ludwig-Maximilians-Universität
Munich, Germany
huebler@ifi.lmu.de, kriegel@dbs.ifi.lmu.de

Karsten Borgwardt, Zoubin Ghahramani
University of Cambridge
Department of Engineering
Cambridge, United Kingdom
kmb51@eng.cam.ac.uk, zoubin@eng.cam.ac.uk

## Abstract

*While data mining in chemoinformatics studied graph data with dozens of nodes, systems biology and the Internet are now generating graph data with thousands and millions of nodes. Hence data mining faces the algorithmic challenge of coping with this significant increase in graph size: Classic algorithms for data analysis are often too expensive and too slow on large graphs.*

*While one strategy to overcome this problem is to design novel efficient algorithms, the other is to 'reduce' the size of the large graph by sampling. This is the scope of this paper: We will present novel Metropolis algorithms for sampling a 'representative' small subgraph from the original large graph, with 'representative' describing the requirement that the sample shall preserve crucial graph properties of the original graph. 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

Graphs are ubiquitous: They are used in various application domains to represent objects and their relationships, including fields such as bioinformatics, systems biology, social network analysis and even software engineering. Whereas in the past, graph structures in application domains such as chemoinformatics included dozens of nodes, nowadays, bioinformatics and the Internet are generating graphs with thousands or even tens and hundreds of thousands of nodes.

This increase in graph size is a challenge for data mining: In many applications we either need to run expensive algorithms such as simulations (routing protocols, virus propagation, 'viral marketing' analysis) on these graphs, or we want to get an impression of the graph topology from visualization. The runtime effort for all these tasks usually scales at least polynomially in the size of the graph, i.e. its number of nodes $n$. For large graphs, these runtimes of $O(n^c)$, where often $c \geq 3$, are not affordable. As we are highly interested in data analysis on these large graphs, we have to work around this problem. Strategy one is to develop scalable algorithms that improve on the $c$ term such that the overall runtime is better than quadratic. This usually comes at the price of employing heuristics which do not guarantee an optimal solution to the problem at hand. Strategy two would be to improve the $n$ term in the runtime effort. Reducing $n$ is equivalent to reducing the graph size. One way to do this is by sampling a subgraph from the large graph such that this subgraph approximates the original graph well. This is the problem we study in this article.

Why could such a subgraph sample be of interest to us? First, we can perform simulations on this sample graph that are too expensive on the large graph. Second, the sample graph may be deemed a 'model' of the large graph, representing its properties in a compact manner. Third, choosing a subgraph sample might provide interesting insights into the nodes that belong to this sample; they are representative of the graph as a whole. Fourth, sampling a smaller subgraph is more 'true' to the data than generating an artificial small graph that approximates the original graph. Any observations we make on the subgraph sample can be pinpointed to a particular node or set of nodes in the original graph. For a generated graph, there is no 1:1 correspondence to nodes in the original graph.

But how does one measure if a graph sample is 'good' and approximates the original graph well? How do we measure if it is a 'representative' sample of the large graph? The idea is to find a subgraph sample $S$ that approximates topological properties $S$ of the original graph $G$, such as its degree distribution. This problem of finding a 'representative subgraph sample' can be cast into the following optimization problem:

$$\text{argmin}_{S \sqsubseteq G \wedge |S| = n'} \Delta(\sigma(S), \sigma(G)) \tag{1}$$

where $G$ is the original graph, $S$ is a subgraph of $G$ with $|S| = n'$ nodes, $\sigma(X)$ is a topological property of graph $X$, and $\Delta$ is a distance function on these topological properties.

For most graph properties and distance functions, solving problem (1) optimally is intractable. We demonstrate this by showing that problem (1) is closely related to a NP-complete problem which we refer to as FIT. We define the FIT problem as

$$\operatorname{argmin}_{S \sqsubseteq G \wedge |S|=n'} \Delta(\sigma(S), \sigma_1) \qquad (2)$$

for a given vector $\sigma_1$.

**Theorem 1** *FIT is NP-complete.*

**Proof** We show that CLIQUE $\leq_p$ FIT. CLIQUE is the problem to decide whether a graph $G$ contains a clique of $n'$ nodes. Searching for a size $n'$-clique in a graph is equivalent to solving the following instance of the FIT problem: Assume $\sigma(S)$ is the degree distribution of subgraph $S$ (normalized to 1) and $\sigma_1$ is equivalent to the degree distribution of a graph whose nodes all have the same degree $n' - 1$ (normalized to 1 as well). If FIT finds a subgraph $S$ such that $\Delta(\sigma(S), \sigma_1) = 0$, then $G$ obviously contains a clique of size $n'$. If $\min \Delta(\sigma(S), \sigma_1) > 0$, then $G$ does not contain a clique of size $n'$. Hence CLIQUE is polynomial-time reducible to FIT, and thus FIT is NP-complete. ■

To cope with the NP-completeness of the FIT problem, we define and explore the use of Metropolis-based sampling and optimization techniques to find 'good' solutions to (1) as measured by $\Delta(\sigma(S), \sigma(G))$.

Another question we are interested in is whether capturing $\sigma(G)$ for a number of key properties $\sigma$ will result in a subgraph $S$ which might embody many *other* properties of $G$. Empirically, we observe, for instance, that by matching the degree distribution of $G$ and $S$, we can preserve other interesting graph properties.

Note that, as commonly done in the graph mining literature, we are dealing with undirected graphs without self-loops and multiple edges here. The samples of our methods always represent *induced* subgraphs of the original graph, that is we keep all edges between nodes in the sample subgraph that are present in the original graph. Methods from the literature that we compare to may generate non-induced subgraphs as well.

## 1.1 Related work

Unlike our approach, existing graph sampling algorithms do not compute properties of the original graph for the actual sampling step. They can be classified into three conceptual categories [9]: Random node selection, random edge selection and sampling by exploration.

There are slightly different variants of randomly selecting nodes (*RandomNode*, *RandomPageRank*, *RandomDegreeeNode*) or edges (*RandomEdge*, *RandomNodeEdge*). For comparison purposes, in this paper, we use the most common instances, namely uniformly distributed settings of *RandomNode* ($RN$) and *RandomEdge* ($RE$). Both algorithms generate induced subgraphs of the original graph from a set of randomly selected nodes or edges.

In contrast, sampling algorithms that perform a graph exploration randomly select a starting node and then visit nodes and edges in its vicinity. The spectrum of explorative algorithms reaches from *RandomNextNeighbor* over *RandomWalk* and *RandomJump* to *ForestFire* exploration techniques. As *ForestFire* ($FF$) with an appropriate forward burning probability $p_f$ has been reported to perform best among all explorative algorithms [9], we use it for comparison to our novel approach, both in an induced ($FF_i$) and not-induced fashion ($FF$). Beginning with a randomly picked seed node *ForestFire* is recursively "burning" a geometrically distributed number of outgoing links together with the corresponding neighbor-nodes. Any "burning" neighbor itself burns a random number (mean: $(p_f/(1 - p_f))$ of its own links. This procedure proceeds until enough ($n'$) nodes are burned. Not-induced *ForestFire* is adding all burned nodes and edges to the sample, while induced *ForestFire* is constructing an induced subgraph using the set of burned nodes. Thus induced *ForestFire* leads to samples being more connected.

In previous work also attempts were made to sample graphs by reduction. Instead of selecting nodes or edges that will be included in the future sample, nodes or edges are deleted from the original graph. Thus the original graph is shrunken to a sample. The number of nodes can be reduced by as much as 70% while preserving important graph properties [8]. Furthermore sampling schemes were developed and investigated for visualization [15]. Some visualization-approaches are compressing the large graph based on a precomputed ranking of vertices [4]. Repeatedly constructive graph modeling approaches are also proposed to be used as sampling algorithms by aborting graph growth ahead of time [10].

## 1.2 Contributions

Whereas existing graph sampling algorithms do not compute properties of the original graph $G$, the key idea of our approach is to use graph properties of the original graph $G$ that are efficient to compute or to approximate to guide us to 'good' representative subgraph samples. Towards this end, we design graph sampling strategies based on the Metropolis Algorithm and Simulated Annealing.

Empirically, our approach shows two remarkable advantages over existing methods:

- First, it generates subgraph samples of **high quality**: The generated sample approximates the properties of the original graph better than previous approaches.
- Second, it allows to find high quality samples of **small size**: The size of our samples is much smaller than those commonly used in the literature.

This article is structured as follows. In Section 2, we review the Metropolis Algorithm and Simulated Annealing, which are the foundations of our approach to representative subgraph sampling. In Section 3, we present our three novel strategies to subgraph sampling. In Section 4, we discuss criteria for assessing the quality of a graph sample. We evaluate the practical performance of all proposed methods in Section 5, before concluding with a short discussion of our findings in Section 6.

## 2 Markov Chain Monte Carlo Methods

From a statistical point of view, *induced* graph sampling is the task to draw a set $S$ of $n'$ nodes from the $n := |V|$ nodes of the original graph $G = (V, E)$. For example in the case of *RandomNode* the samples $S$ are uniformly distributed on the sample space $\mathfrak{X} = \{S \sqsubseteq V | n' = |S|\}$. By contrast the distribution of better performing explorative sampling algorithms like *induced ForestFire* is not explicitly known.

The main idea of Metropolis graph sampling is to draw a sample from the sample space $\mathfrak{X}$ following a specific density $\varrho(S)$. This density should reflect subgraph sample quality well, which means good induced samples $S$ should be drawn more frequently than worse ones. Thus $\varrho(S)$ depends on the quality of the sample $S$.

From this point of view an obvious choice of $\varrho(S)$ is one depending on a distance measure with respect to a preprocessed graph property. The severe problem with such a density $\varrho(S)$ is that it is only given in an unnormalized manner $\varrho^*(S)$. To obtain the normalized density $\varrho(S)$ we have to calculate and sum over $\binom{n}{n'}$ summands which is obviously intractable:

$$\varrho(S) = \frac{\varrho^*(S)}{\sum_{S' \in \mathfrak{X}} \varrho^*(S')}$$

How can we draw samples from the sample space $\mathfrak{X}$ if the underlying normalized density $\varrho(S)$ is not given explicitly? This problem is solved by the Metropolis algorithm.

### 2.1 Metropolis algorithm

The Metropolis Algorithm [11, 5] can draw samples from any probability distribution, provided that a function $\varrho*$ proportional to the density $\varrho(S)$ can be calculated for any $S \in \mathfrak{X}$.

Such a stochastic simulation is reached by using Markov Chains. A Markov Chain is a stochastic process with the property that future states depend only on the current state but not on past states. In our case each state depicts a single set of nodes $S$ which in turn represents a sample $S$ with n' nodes. In general Markov Chains are defined as follows:

**Definition 2 (Markov Chain)** *Let $\Pi$ denote a stochastic matrix. Let $(X_n)_{n\geq 0}$ be a sequence of random variables operating on a probability space $(\Omega, \mathcal{F}, P)$ with values in $\mathfrak{X}$. The sequence $(X_n)_{n\geq 0}$ is called Markov Chain with state-space $\mathfrak{X}$ and transition matrix $\Pi$, if $\forall S_0, \ldots, S_{n+1} \in \mathfrak{X}$ with $P(X_0 = S_0, \ldots, X_n = S_n) > 0$:*

$$P(X_{n+1} = S_{n+1} | X_0 = S_0, \ldots, X_n = S_n) = \Pi(S_n, S_{n+1})$$

For the Metropolis algorithm the central property of Markov Chains is their ergodicity under certain conditions [13]. The ergodicity-theorem states that the Markov Chain converges through transitions to a stationary distribution if and only if any state is reachable by any other state in a finite number of transitions.

This property is used alongside with *detailed balance*

$$\varrho(S)\Pi(S, S') = \varrho(S')\Pi(S', S)$$

to obtain a Markov Chain with equilibrium distribution $\varrho(S)$. Therefore we can simulate transitions on this Markov Chain until it converges and then draw a sample which is distributed according to $\varrho(S)$. The clue is that the unnormalized density $\varrho^*(S)$ suffices to the simulation of the Markov Chain because it is proportional to $\varrho(S)$ and the normalizing constant can be canceled out by detailed balance.

In the basic Metropolis algorithm, the transition probability $\Pi(S, S')$ is separated into a proposal distribution $Q(S, S')$, describing the probability of proposing a move to state $S'$ from state $S$, and an acceptance probability $a(S, S')$, thus $\Pi(S, S') = Q(S, S') \cdot a(S, S')$ with $S \neq S'$. The acceptance probability has to be chosen in such a way that detailed balance is still ensured: $a(S, S') = \min(1, \frac{Q(S,S')}{Q(S',S)} \frac{\varrho^*(S')}{\varrho^*(S)})$. The proposal distribution $Q(S, S')$ can be easy to compute, for instance, it can be uniformly distributed over some set of states $S'$. Thus instead of calculating the probability of any possible transition the algorithm only needs to calculate the acceptance probabilities of transitions which are actually proposed. A non-uniform proposal distribution can be used to preferentially propose certain transitions. In this article we always utilize symmetric proposal distributions $Q(S, S') = Q(S', S)$. Hence the acceptance probability is simplified to: $a(S, S') = \min(1, \frac{\varrho^*(S')}{\varrho^*(S)})$.

The following proposition proves our approach to be accurate:

**Proposition 3** *Assume that the constructed Markov Chain only has transitions between adjacent states (displaying adjacent subgraphs) using a symmetric proposal distribution and the above acceptance probability. Under these conditions, the equilibrium distribution of the Markov Chain is exactly the desired density $\varrho(S)$.*

The proof of this proposition is straigthforward and due to space limitations only shown in the appendix (`http://mlg.eng.cam.ac.uk/~karsten/RSS_ICDM08/icdm_appendix.pdf`).

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 $\varrho^*(S)$, which in our case is inversely proportional to a distance measure $\Delta(\sigma(G), \sigma(S)) = \Delta_{G,\sigma}(S)$ between the real graph $G$ and the sample $S$.

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 $\Delta_{G,\sigma}(S)$ by a large positive scalar $p$. Hence we define $\varrho^*$ as

$$\varrho^*(S) := \frac{1}{\Delta_{G,\sigma}(S)^p} = \frac{1}{\Delta(\sigma(G), \sigma(S))^p}, \tag{3}$$

where $p \in \mathbb{R}^+$ and $p \gg 0$.

### 2.2 Simulated Annealing

The problem of Metropolis is that the convergence of the Markov Chain can be slowed down enormously if the Markov Chain gets stuck in a local maximum of the underlying density $\varrho(S)$. If the Markov Chain enters such a state it is unlikely that this state is left within a reasonable amount of time. An approach to solve this problem is Simulated Annealing [7]: It is based on the idea that states with high density $\varrho(S)$ (called low energy states) are mostly not uniformly distributed on the state space $\mathfrak{X}$. Thus we change the density $\varrho(S)$ in such a way that in the beginning, more transitions are accepted (high temperature) and in the very end, only transitions are accepted which are an improvement (low temperature). Thus energy barriers arising from regions of low density separating regions of high density can be overcome and local energy minimums (maxima of density) can be left in time. The hope is that in the end the simulation ends up on the right side of the barrier where the global maximum or at least the better local maximum can be found. This approach is based on the idea that the distribution at higher temperature is a good guide to the distribution at lower temperature. We define the temperature-dependent density as

$$\varrho^*_{p,T}(S) = e^{\frac{\log \varrho^*_p(S)}{T}} = e^{log\Delta_{G,\sigma}(S)^{(-\frac{p}{T})}} = \Delta_{G,\sigma}(S)^{(-\frac{p}{T})}$$

so that a new acceptance-probability follows

$$a(S, S') = min(1, \frac{\varrho^*_{p,T}(S')}{\varrho^*_{p,T}(S)}) = min(1, (\frac{\Delta_{G,\sigma}(S)}{\Delta_{G,\sigma}(S')})^{\frac{p}{T}})$$

assuming a symmetric proposal distribution $Q(S, S') = Q(S', S)$.

For our experiments, we use Simulated Annealing with a geometric annealing schedule, which means that we gradually reduce the temperature according to $T_{t+1} = \gamma T_t$ with $0 < \gamma < 1$, where $t$ is the $t$-th step in our sampling procedure.

## 3 Metropolis Graph Sampling

We use the Metropolis algorithm in order to optimize a randomly picked initial sample. The key idea is that after picking a random subgraph sample $S$ from $G$, we search the space of subgraphs by removing a node from and adding a new node to $S$ in each iteration.

### 3.1 Adaptation to graph sampling

As stated earlier, we choose a $\varrho^*(S)$ that is inversely proportional to $\Delta_{G,\sigma}(S)^p$. $\Delta_{G,\sigma}(S)$ can be any distance measure between topological properties $\sigma$ of the sample $S$ and the original graph $G$, for instance, a distance on degree distributions. We discuss these graph properties in Section 4.1 and suitable distance measures in Section 4.2. On the one hand, these graph properties help us to find a representative subgraph sample. On the other hand, we have to compute these properties for the original graph $G$ as well as for every sample visited by the Markov Chain. For this reason, we restrict ourselves to graph properties that are efficient to compute (like e.g. the degree distribution). The pseudocode of our Metropolis Graph Sampling algorithm (without annealing) is depicted in Algorithm 1.

### 3.2 Complexity analysis

There are 4 crucial steps in our Metropolis subgraph sampling: First, the time effort to read the graph $G$, which is $O(n \cdot d_{avg})$, where $d_{avg}$ is the average degree of a node in $G$; second, the computation time for the graph property $\sigma(G)$, denoted by $R(\sigma(G))$; third, the time effort to pick an initial sample, which is in $O(n' \cdot d_{avg})$; fourth, the runtime effort for each of the $\#it$ iterations of the Markov Chain, which includes updating our subgraph sample (removing one node, adding one node) in $O(n' \cdot d_{avg})$ and computing its graph properties in $R(\sigma(S_{current}))$. Hence the over-all complexity is

$$O(n \cdot d_{avg} + R(\sigma(G)) + \#it\,(d_{avg} \cdot n' + R(\sigma(S_{current})))).$$

**Algorithm 1** Metropolis Subgraph Sampling

```
Input: Graph $G = (V, E)$, distance function $\Delta_{G,\sigma}(\cdot)$,
    sample size $n'$, number of possible transitions $\#it$, exponent $p$

  $S_{current} \leftarrow$ uniformly at random from $G$
  $S_{best} := S_{current}$
  for $i := 1$ to $\#it$ do
    node $v \leftarrow$ randomly from $S_{current}$
    node $w \leftarrow$ randomly from $(V \setminus S_{current}) \cup \{v\}$
    $S_{new} := (S_{current} \setminus \{v\}) \cup \{w\}$
    $\alpha \leftarrow$ uniformly at random from interval $[0, 1]$
    if $\alpha < (\frac{\Delta_{G,\sigma}(S_{current})}{\Delta_{G,\sigma}(S_{new})})^p$ then
      $S_{current} := S_{new}$
      if $\Delta_{G,\sigma}(S_{current}) < \Delta_{G,\sigma}(S_{best})$ then
        $S_{best} := S_{current}$
      end if
    end if
  end for

Output: $S_{best}$
```

## 3.3 Choice of the exponent $p$

The choice of the exponent $p$ depends on the original graph $G$. As $p$ is responsible for good samples being favored over worse ones, it very much depends on the real graph $G$ and the distance measure $\Delta$. In principle, any $p > 0$ will result in the same optima, however larger $p$ create a more peaked probability distribution.

If $G$ has a high fraction of medium-quality samples then $p$ needs to be of high value to make the small group of extraordinary samples competitive. Most graph properties used as similarity measures evaluating a sample $S$, firstly depend on whether $S$ is connected. Because of that a suitable $p$ is depending on the edges-per-node ratio of $G$ which determines the total number of possible connected subgraphs. Besides graphs of large size also demand high values of $p$ as the overall fraction of good samples shrinks with the graph size $n$. We use the following rule of thumb in all of our experiments (with $k$ being the number of edges of $G$): $p_G = 10 \cdot \frac{k}{n} \log_{10} n$.

Why do we not generally use either enormous $p$ values or an exponential function of $\Delta_{G,\sigma}(S)$? To answer this question, one should bear in mind that large differences in density are seldom overcome by the Markov Chain. Large or exponential choice of $p$ might enhance convergence, but one then faces an increased risk of getting stuck in a state which is a local maximum of the density.

## 3.4 Speeding up convergence

As we use Metropolis for optimizing (as described in 2.1), convergence of the Markov Chain is equivalent to the achievement of an appropriate sample. In this section we discuss a common and a new graph specific approach to decrease the number of required iterations needed to achieve convergence of the Markov Chain and thus a good sample. By decreasing the number of performed iterations we hope to notedly reduce the CPU runtime.

**Simulated Annealing in graph setting** A common way to avoid that the Metropolis algorithm gets stuck in local optima is to use Simulated Annealing. Since local optima can slow down convergence, Simulated Annealing may be useful for speeding up the sampling process.

**Chaining** 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 the idea arises of restricting the search space $\mathfrak{X}$ to connected samples. Let us assume for all of the following results that the original graph $G$ is connected. If $G$ has more than 1 component, the following sampling algorithm is still applicable, but will sample a subgraph within the component of the initial sample only.

Restricting ourselves to connected samples shrinks the state space of the Markov Chain, so we have to adapt the proposal distribution $Q(S, S')$, so that detailed balance is achieved. We do this by initially deleting a randomly selected node from $S$. The reduced sample is called $S_r$. Then we add another random node from $N(S_r) = \{v \in V(G) | v \notin V(S_r) \wedge \exists w \in V(S_r) \text{s.t. } e = (v, w) \in E(G)\}$ of adjacent nodes to the reduced set of nodes $S_r$. If the resulting induced subgraph $S'$ is not connected because a bridge node — whose removal disconnects the sample — was deleted, we reenter the old state $S$. Otherwise we decide according to the acceptance probability $a(S, S')$ whether we should pass into the new state $S'$.

In contrast to original Metropolis graph sampling all states of Chaining's reduced state space are of a minimum quality because of being connected. Thus less proposed transitions are rejected due to higher values of $a(S, S')$ in average. For this reason the hope is that the amount of necessary iterations to achieve convergence can be decreased.

We want to show that the Markov Chain with restricted state space $\mathfrak{X}_{con}$ converges to $\varrho(S) = \frac{\varrho^*(S)}{\sum_{S' \in \mathfrak{X}_{con}} \varrho^*(S')}$, under the assumption that the original graph $G$ is connected. The proof of proposition 3 can be directly adopted if we show the restricted proposal distribution to be symmetric:

**Proposition 4** *For the Chaining algorithm's construction of new states (as described above) it holds that* $Q(S, S') = Q(S', S)$.

**Proof** For $S = S'$ or $|S \cap S'| < n' - 1$ this is trivial. Otherwise $S$ and $S'$ are adjacent. The proposal probabilities $Q(S, S')$ and $Q(S', S)$ describe the cases of constructing $S$ out of $S'$ and $S'$ out of $S$ respectively. In both cases first of all we delete a randomly drawn node out of the starting set of nodes ($S$ or $S'$). The probability of picking exactly the node $S$ and $S'$ differ in is $(\frac{1}{n'})$. This probability is independent of the direction of the transition. The resulting graph $S \cap S'$ containing $n' - 1$ nodes is equal in both cases. Thus the probability of constructing $S$ and $S'$ respectively out of $S \cap S'$ is $1/N(S \cap S')$. Hence $Q(S, S') = \frac{1}{n'} \cdot \frac{1}{N(S \cap S')} = Q(S', S)$. ■

As stated earlier (proposition 3) with this requirement of symmetry detailed balance, the ergodicity-theorem and thus the stationary distribution $\varrho$ can be followed.

It has to be mentioned that the reduction of the state space $\mathfrak{X}$ to $\mathfrak{X}_{con}$ leads to occasional construction of illegal unconnected subgraphs by deletion of bridge nodes. Thus connectivity has to be checked and if necessary transitions have to be reversed. The connectivity-test of the new sample is performed by depth-first-search requiring $O(n' + k') = O(n' \cdot d_{avg})$ additional operation time per transition. Furthermore a list of adjacent nodes $N(x)$ has to be set up in $O(n' \cdot d_{avg})$ time. The updating of this neighborhood takes $O(d_{avg})$. Thus the complexity of Chaining in addition to mere Metropolis is $O(n' \cdot d_{avg} + \#it \cdot n' \cdot d_{avg})$. In particular, Chaining is attractive when graph properties of the subgraph sample are expensive to compute, as Chaining decreases the overall number of iterations.

## 4 Sample Evaluation

Before testing our sampling algorithm, we have to answer an important question: Which graph properties do we consider and which distance or similarity measures do we employ to measure the discrepancy between properties of the sample graph and the original graph, and hence the 'quality' of the subgraph sample?

It is important to bear the following in mind: Both our graph properties and the similarity measures on them can be used i) inside our subgraph sampling algorithms to guide us to 'better' subgraphs (*usage for sampling*) and ii) to evaluate afterwards if the generated subgraph preserves properties of the original graph well (*usage for evaluation*). Note that we may use one property $A$ for sampling, but employ a different property $B$ for evaluation of the subgraph sample (as done in Section 5).

### 4.1 Graph properties

We used the following graph properties to guide our Metropolis algorithms to find a solution to problem (1).

A well-known graph property is a graph's **degree distribution**, with the degree of a node being the number of its incident edges. Its popularity is certainly derived partly from the fact that the degree distribution can be obtained in $O(n \cdot d_{avg})$ using adjacency lists, which is not more expensive than the runtime effort for simply reading the graph.

The **clustering coefficient** $C_v$ of a node $v$ with degree $d(v)$ is defined as the number of edges actually existing between neighbors of $v$ divided by the maximum possible number of such edges between its neighbors, $d(v)(d(v) - 1)/2$. We represent the clustering coefficients of a graph in terms of a vector with values $C_d$ (called clustering coefficient 'distribution' in [9]), where $C_d$ is defined as the average $C_v$ over all nodes $v$ of degree $d$. The worst-case complexity of determining this clustering coefficient vector is $O(n^3)$, but can be determined efficiently on the real-world sparse graphs in our experiments.

Besides these well-known topological properties, we also consider the **graphlet distribution** of our graphs [14, 12]. Given a graph $G$, a $k$-graphlet is a connected and induced subgraph of $G$ of size $k$. We use the distribution of 3-, 4-, and 5-graphlets in our graphs. As a sparse graph has $O((d_{max})^{k-1})$ graphlets [6] (with $d_{max}$ being the maximum degree), we have to resort to sampling if we want to determine the graphlet distribution efficiently. Towards this end, we use a subgraph sampling algorithm by Wernicke [17].

We use the aforementioned properties both for sampling subgraphs and for evaluating the quality of the produced sample afterwards. In addition, we used the following graph properties exclusively for evaluation.

The **diameter** of a graph is the maximum shortest path length between any pair of nodes in the graph, computable in a worst-case runtime of $O(n^3)$.

### 4.2 Distance functions on graph properties

A proper way of measuring similarity between properties of our original graph and its subgraph sample is a key component of our approach and graph sampling in general. Towards this end, we employ the following distance function, with $\mathcal{M}$ denoting the universe of the particular graph property, for instance the set of all existing types of graphlets, or the set of all existing node degrees. Let $g = \sigma(G)$ denote the properties of the original graph, and $s = \sigma(S)$ the properties of the subgraph sample in the following.

**Graphlets:** Distributions with nominal measurement scale like the graphlet distribution of the graph ($g$) and sample ($s$) are compared using a double sided version of the well known entropy-distance: $\Delta_1(g, s) = \sum_{i \in \mathcal{M}} \frac{(g(i) - s(i))^2}{g(i) + s(i)}$. In our variant of this distance function, we add the extra term $s(i)$ to the denominator. The reason is that in certain existing graph sampling algorithms (such as *RandomEdge*) we are sampling *non-induced* subgraphs from a graph. For this reason, it may happen that the sample contains graphlets which do not appear in the original graph, as they are only *non-induced* subgraphs in the original graph, but *induced* subgraphs in the sample. The extra term $s(i)$ then avoids division by zero. In the literature of graphlets [14] another distance measure has been proposed: $\Delta(g,s) = \sum_{i \in \mathcal{M}} |\log \frac{g(i)}{s(i)}|$ We do not use this measure because of several weaknesses: First of all in the above case of the sample containing a graphlet not existing in the original graph the measure is undefined. Second the measure is infinite if any sort of graphlet occurring in the graph does not exist in the sample, which often occurs.

**Degree distribution:** We compare distributions with at least ordinal measurement scale, such as the degree distribution, using the Kolmogorov-Smirnov D-Statistics, which corresponds to the maximum difference between the two cumulative distribution functions $F_Y$ of $g$ and $F_{Y'}$ of $s$ over the range of the random variables $Y$ and $Y'$ on $\mathcal{M}$. $Y$ and $Y'$ are distributed according to $g$ and $s$ respectively: $\Delta_2(g,s) = max_{i \in \mathcal{M}} |F_Y(i) - F_{Y'}(i)| = max_{i \in \mathcal{M}} |P(Y \leq i) - P(Y' \leq i)|$. It is derived from the Kolmogorov-Smirnov test.

**Clustering coefficient:** The clustering coefficient (as defined here) is a vector with values in $[0,1]$ and therefore will be compared using the $L_1$-norm. Higher L-norms including $L_\infty$ are not suitable for our purposes because the coefficient of a single degree should not dominate the distance. As we normalise the possible distances to $[0,1]$, we are in fact calculating the average (absolute) difference in clustering coefficients over all degrees: $\Delta_3(g,s) = \frac{\sum_{i \in \mathcal{M}} |g(i) - s(i)|}{|\mathcal{M}|}$.

**Diameter:** The diameter of the sample graph is measured as percent deviation from the original diameter. Thus the sample-diameter can take values in $[0,1]$ and therefore is comparable to the above distance measures.

## 5 Experimental evaluation

### 5.1 Datasets

We test our described algorithms on the following 5 datasets, each of them representing a single graph: **Dobson&Doig** ($DD_{1BK0}$), which is a graph model of a protein [3]; **Autonomous systems** ($AS_{NOV97}$), which is a graph model of the Internet [2]; a **Network of trust** ($Epinions$) from epinions.com, describing who trusts whom in a social network; the yeast **PPI network** ($yeast20071104$) which is a network of protein interactions [16]; a **Citation network** ($HEP - PH$) describing who cites whom in the discipline of high energy physics [1]. Crucial statistics of these datasets are shown in Table 5.1, more detailed descriptions can be found in the appendix.

### 5.2 Experimental setting

**Sampling** On these 5 datasets, we ran 15 different sampling strategies to generate a representative subgraph sample with $n' = 100$ nodes each.[1]

[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.

| dataset | $n$ | $k$ | $d_{avg}$ | $d_{max}$ | $comp$ |
|---|---|---|---|---|---|
| $DD_{1BK0}$ | 329 | 2,100 | 12 | 22 | 1 |
| $AS_{NOV97}$ | 3,082 | 5,281 | 3 | 600 | 1 |
| $Epinions$ | 75,879 | 405,801 | 10 | 3,044 | 2 |
| $yeast20071104$ | 4,932 | 17,346 | 7 | 283 | 32 |
| $HEP - PH$ | 34,546 | 420,899 | 24 | 846 | 61 |

**Table 1.** Statistics on the real-world graphs used in our experiments ($n$ = number of nodes, $k$ = number of edges, $d_{avg}$ = average degree, $d_{max}$ = maximum degree, $comp$ = number of connected components).

These 15 approaches included 4 state-of-the-art methods for representative subgraph sampling (see Section 1.1): RandomNode ($RN$) and RandomEdge ($RE$), that randomly sample nodes and edges from the graph, and Forest-Fire, once for sampling induced subgraphs ($FF_i$), once for sampling non-induced subgraphs ($FF$).

We compared these state-of-the-art 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 (described in Section 5.3).

We set $\varrho^*$ as in equation (3) and determined the exponent $p$ by the rule of thumb from Section 3.3. 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 $Ch_{10dcg}^{SA}$).

We repeat each experiment 25 times to avoid random effects.

**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 ($graphlet$), employing the distance function from Section 4.2. In addition, we compute the mean of these 4 distances, which gives us the average distance, denoted by $AVG$. We also show the empirical standard deviation of the average distance ($SD$). We report sample quality in terms of these distances between the subgraph sample and the original graph in Table 2 as averages over 25 repetitions on all 5 datasets. Results for individual datasets and variance of sample quality are shown in the Appendix.

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 and to set up its adjacency list. $t_{run}$ is the runtime of the actual sampling stage.

## 5.3 Parameter settings

Before we discuss our results, we provide the crucial parameter settings 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 [9] 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 $p_f = 0.7$ (performing best in these test runs).

**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 $p = 10 \cdot \frac{k}{n} \log_{10} n$, as described in Section 3.3.

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 (see Section 5.4). 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.

## 5.4 Results

**Overall sample quality** As can be seen from Table 2 (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. We further investigated the empirical standard deviation (column SD) and observed that results of Metropolis sampling (including its variants Simulated Annealing and Chaining) on average show less variance than those of existing subgraph sampling algorithms.

**Metropolis** On 4 out of 5 of our datasets, Metropolis sampling approximating the degree sequence ($M_d$) outperforms all existing sampling algorithms. Dataset specific results are shown in the appendix. The only dataset on which our methods do not perform excellent is HEP-PH. The problem for Metropolis is that the sample size is close to the number of components on HEP (64 components vs. 100 nodes in sample), so samples spreading their nodes uniformly over different components cannot really capture graph properties well, and are hence rejected. All sampling methods have similar problems with this dataset, but we still ran tests on it, as it was used in the pioneering paper on subgraph sampling [9].

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, we make two important observations: On the one hand, our methods require more runtime than those in the literature, because they are computing properties of the original graph ($t_{prep}$) and of the subgraph samples during sampling (part of $t_{run}$). On the other hand, our methods still exhibit attractive runtimes: Even on the largest graph (Epinions, 75,879 nodes), our slowest method ($M_{10dcg}$) generates a high-quality subgraph sample in roughly 367 seconds. Our fastest method $M_d^{SA}$ generates a high-quality sample in 2 seconds on the same dataset.

There are interesting differences in runtime between different variants of our sampling approach. $M_d$ 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 than the other graph properties.

**Chaining** Chaining leads to a speed-up in runtime if its computational overhead, that is 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 (for instance, a weighted combination of several properties such as in $Ch_{10dcg}$), but not for properties such as the degree distribution that are efficient 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. Still, it outperforms on average all state-of-the-art methods with respect to sample quality without the need to pick an exponent $p$.

**Variation of exponent** $p$ For Metropolis and Chaining in its standard version, the exponent $p$ is a central parameter. To assess its influence on sample quality, we vary $p$ from 50 to 1,000 in steps of 50, and run $M_d$ on the Epinions dataset using these values of $p$ (and in addition for $p = 25$). We achieve the smallest distance to the original graph (as

| | degree | diam | clust | graphlet | AVG | SD | $t_{read}$ | $t_{prep}$ | $t_{run}$ | $t$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $RN$ | 0.911 | 0.653 | 0.332 | 0.540 | 0.487 | 0.068 | 0.491 | 0.000 | 0.008 | 0.499 |
| $RE$ | 0.725 | 0.633 | 0.389 | 0.500 | 0.449 | 0.066 | 0.491 | 0.000 | 0.009 | 0.500 |
| $FF$ | 0.450 | 0.455 | 0.311 | 0.268 | 0.297 | 0.015 | 0.491 | 0.000 | 0.004 | 0.495 |
| $FF_i$ | 0.340 | 0.157 | 0.132 | 0.280 | 0.227 | 0.030 | 0.491 | 0.000 | 0.012 | 0.503 |
| $M_d$ | **0.105** | 0.247 | 0.121 | 0.143 | 0.154 | 0.015 | 0.491 | 0.001 | 3.096 | 3.587 |
| $M_c$ | 0.814 | 0.498 | 0.072 | 0.287 | 0.418 | 0.052 | 0.491 | 5.227 | 2.857 | 8.575 |
| $M_g$ | 0.788 | 0.531 | 0.405 | 0.177 | 0.475 | 0.026 | 0.491 | 64.572 | 4.430 | 69.493 |
| $M_{dcg}$ | 0.544 | 0.326 | 0.069 | 0.072 | 0.253 | 0.019 | 0.491 | 69.237 | 35.088 | 104.816 |
| $M_{10dcg}$ | 0.164 | 0.220 | **0.063** | 0.084 | 0.132 | 0.015 | 0.491 | 62.365 | 110.277 | 173.133 |
| $M_d^{SA}$ | 0.147 | 0.252 | 0.121 | 0.156 | 0.169 | 0.014 | 0.491 | 0.001 | 2.613 | 3.104 |
| $M_{10dcg}^{SA}$ | 0.214 | 0.256 | 0.067 | 0.072 | 0.147 | 0.016 | 0.491 | 69.173 | 95.761 | 165.425 |
| $Ch_d$ | 0.182 | 0.172 | 0.119 | 0.077 | 0.137 | **0.013** | 0.491 | 0.001 | 8.965 | 9.456 |
| $Ch_{10dcg}$ | 0.189 | 0.149 | 0.087 | 0.090 | **0.129** | 0.017 | 0.491 | 63.073 | 37.257 | 100.821 |
| $Ch_d^{SA}$ | 0.204 | **0.139** | 0.139 | 0.086 | 0.142 | 0.018 | 0.491 | 0.000 | 8.830 | 9.321 |
| $Ch_{10dcg}^{SA}$ | 0.221 | 0.181 | 0.111 | **0.065** | 0.148 | 0.019 | 0.491 | 62.993 | 32.923 | 96.407 |

**Table 2.** 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). All data shown is the average over 25 repetitions on all 5 datasets. For full details on all abbreviations see Section 5.2.

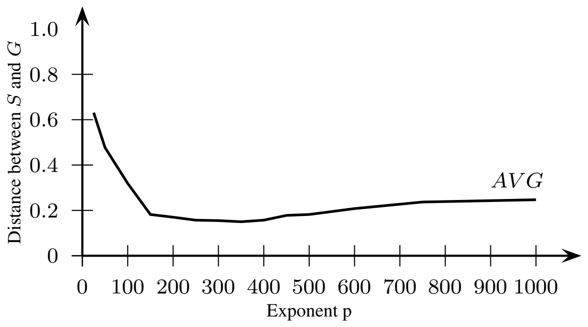

**Figure 1.** Distance between sample graph and original graph $Epinions$ versus size of exponent $p$ (Metropolis Sampling $M_d$; distance is measured in terms of average over four graph properties from Section 4.1).

| $\frac{n'}{n}$ in % | 5 | 10 | 20 | 30 | 50 | 75 |
|---|---|---|---|---|---|---|
| $\Delta_d(FF_i)$ | 0.225 | 0.228 | 0.187 | 0.150 | 0.118 | 0.060 |
| $\Delta_d(M_d)$ | 0.006 | 0.006 | 0.005 | 0.004 | 0.002 | 0.001 |
| $t(FF_i)$ | 0.508 | 0.514 | 0.538 | 0.639 | 0.825 | 1.506 |
| $t(M_d)$ | 3.296 | 5.741 | 16.626 | 22.374 | 40.960 | 54.366 |

**Table 3.** Distance between sample and original graph $AS_{NOV97}$ in terms of degree distribution $\Delta_d$ for Forest Fire ($FF_i$) and Metropolis ($M_d$), for various sample sizes $n'/n$. Corresponding CPU runtime $t$ is shown in seconds.

an average over several graph properties) for $p = 350$ (see Figure 1). Small choices of $p$ lead to higher distances to the original graph's properties, and choosing larger values of $p$ slowly increases the distance as well.

Our rule of thumb from Section 3.3 apparently makes us pick values of $p$ that let us outperform state-of-the-art methods on our 5 datasets in the vast majority of test runs. Note that if one wants to avoid setting $p$ at all, one may resort to Simulated Annealing.

**Variation of iterations** The impact of a duplication of the number of performed iterations is relatively small and very much depends on the type of investigated graph $G$. Further information is given in the appendix.

**Variation of sample size** So far, we have generated representative subgraph samples with $n' = 100$ nodes, which approximate the original graph extremely well. Can we also achieve these good results for larger subgraph samples?

We study this question on the Autonomous Systems graph ($AS_{NOV97}$) by sampling subgraphs. Table 3 shows the quality with respect to the degree distribution, while Figure 2 shows the quality referring to the average of our four graph properties used for evaluation. For both evaluation criteria (degree and average) and for all different sample sizes, we observe that Metropolis ($M_d$) generates subgraph samples that approximate the original graph's properties better than the samples from induced *ForestFire* ($FF_i$).

A thrilling result of the observations in Table 3 is that samples constructed using Metropolis can outperform samples drawn by *ForestFire* in terms of sample quality, even if they are ten times smaller (Metropolis $M_d$: $\frac{n'}{n} = 0.05$, $\Delta(M_d) = 0.006$; ForestFire: $\frac{n'}{n} = 0.50$, $\Delta(FF_i) = 0.118$). At the same time, for larger samples, our methods

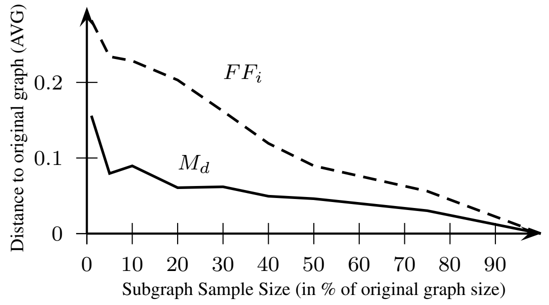

**Figure 2.** Distance between sample graph and original graph G ($AS_{NOV97}$) versus size of sample graph (Distance is measured in terms of an average on the four graph properties from Section 4.1. Sample size is given as ratio $n'/n$).

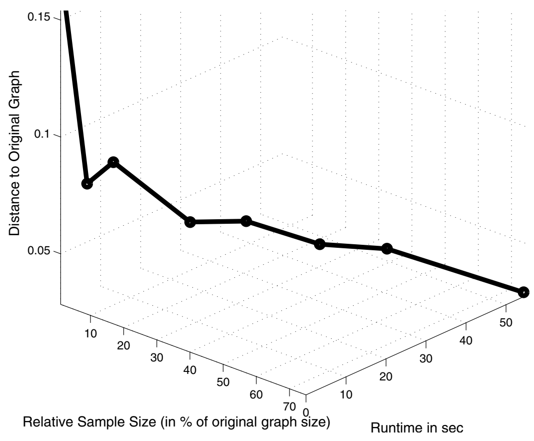

**Figure 3.** Metropolis Sampling: Runtime vs. distance $\Delta_{G,\sigma}(S)$ vs. sample size. Original graph is $AS_{NOV97}$ (Distance is measured in terms of an average on the four graph properties from Section 4.1)

.

require more runtime (see Table 3), as it gets more expensive to compute the graph properties of larger samples and the size of the sample search space increases. Thereby the quality of the sample betters itself as shown in figure 3.

## 6 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 graph $G$, which are then to be approximated well by the sample graph $S$. While existing sampling algorithms only manage to produce good samples with a minimum size of $15\%$ (of nodes of the original graph) [9], our algorithms succeed in constructing representative samples of much smaller size (down to $0.14\%$ on Epinions).

As a practical note: Among the different variants of our approach, Metropolis graph sampling approximating the degree distribution ($M_d$) of the original graph offers the best trade-off between runtime and sample quality. If runtime is less of an issue, the Chaining variant, which approximates degree distribution, clustering coefficient and graphlet distribution of the input graph ($Ch_{10dcg}$), is the best choice.

## References

[1] http://www.cs.cornell.edu/projects/kddcup/datasets.html.

[2] http://www.routeviews.org/.

[3] P. Dobson and A. Doig. Distinguishing enzymes structures from non-enzymes without alignments. In *Journal of Molecular Biology*, volume 330, pages 771–783, 2003.

[4] A. Gilbert and K. Levchenko. Compressing network graphs. In *Proceedings of the LinkKDD workshop at the 10th ACM Conference on KDD*, 2004.

[5] K. Hastings. Equation of state calculations by fast computing machines. In *Biometrika. 57*, pages 97–109, 1970.

[6] S. Itzkovitz, R. Milo, N. Kashtan, G. Ziv, and U. Alon. Subgraphs in random networks. *Physical Rev. E*, 68, 2003.

[7] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. In *Science, Volume 220, Number 4598*, pages 671–680, 1983.

[8] V. Krishnamurthy, M. Faloutsos, M. Chrobak, L. Lao, J. Cui, and A. Percus. Reducing large internet topologies for faster simulations. In *Proceedings of the IFIP Networking, Waterloo, Canada*, 2005.

[9] J. Leskovec and C. Faloutsos. Sampling from large graphs. In *KDD*, 2006.

[10] J. Leskovec and C. Faloutsos. Scalable modeling of real graphs using kronecker multiplication. In *ICML*, pages 497–504, New York, NY, USA, 2007. ACM.

[11] N. Metropolis, A. Rosenbluth, N. Rosenbluth, and A. Teller. Equation of state calculations by fast computing machines. In *The Journal of Chemical Physics, Volume 21, Number 6*, pages 1087–1092, 1953.

[12] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: Simple building blocks of complex networks. *Science*, 298:824–827, 2002.

[13] R. M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, University of Toronto, 1993.

[14] N. Przulj, D. G. Corneil, and I. Jurisica. Modeling interactome: scale-free or geometric? In *Bioinformatics 2004 20(18)*, pages 3508–3515, 2004.

[15] D. Rafiei and S. Curial. Effectively visualizing large networks through sampling. In *Proc. of the IEEE Visualization Conference*, 2005.

[16] L. Salwinski, C. Miller, A. Smith, J. Pettit, F.and Bowie, and D. Eisenberg. The database of interacting proteins: 2004 update. 2004.

[17] S. Wernicke. Efficient detection of network motifs. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 3(4):347–359, 2006.