Dynamic Properties of Network Samples

Bachelor-Thesis von Benedict Norbert Jahn
Juli 2014

TECHNISCHE UNIVERSITÄT DARMSTADT

Department of Computer Science
Peer-to-Peer Networking Group

Dynamic Properties of Network Samples

Vorgelegte Bachelor-Thesis von Benedict Norbert Jahn

1. Gutachten: Benjamin Schiller
2. Gutachten: Prof. Dr. Thorsten Strufe

Tag der Einreichung:

Erklärung zur Bachelor-Thesis

Hiermit versichere ich, die vorliegende Bachelor-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 15. Juli 2014

(B. Jahn)

Abstract

The samplig of social networks has become more and more important in computer since. But network sampling is a broad topic and not every algorithm fits the needs. Common sampling algorithms have different advantages and disadvantages towards the analysis of network samples. The focus of this work is to understand how and why sampling algorithms have different performances in analyzing networks. Our focus lies on the performance towards the Maximum Connected Network Coverage (MCNC) problem. The MCNC problem deals with the challange to explore and see the whole network with as least steps as possible. A node is marked as seen if a neighbor of the node is visited. The MCNC problem is of special interest regarding the growing importance of (online) social networks. Often we do not have detailed informations about those networks and therefore the analysis of sampling algorithms that do not have previous knowledge of the network were on focus of this work. Our analysis shows that the performance of sampling algorithms is highly dependent on the structure of the network. Sampling algorithms like Maximum Observed Degree that tend to visit clusters and highly connected nodes, performs better on social networks, but they do not perform well on technical or non-social networks. Sampling algorithms like the DFS that avoids clustered nodes are not very effective on social networks regarding the MCNC problem, but they have their advantages on random generated and technical networks.

Contents

1 Introduction

The samplig of social networks has become more and more important in computer since. But network sampling is a broad topic and not every algorithm fits the needs. Common sampling algorithms have different advantages and disadvantages towards the analysis of network samples. The focus of this work is to understand how and why sampling algorithms have different performances in analyzing networks. Our focus lies on the performance towards the Maximum Connected Network Coverage (MCNC) problem. The MCNC problem deals with the challange to explore and see the whole network with as least steps as possible. A node is marked as seen if a neighbor of the node is visited. The MCNC problem is of special interest regarding the growing importance of (online) social networks. Often we do not have detailed informations about those networks and therefore the analysis of sampling algorithms that do not have previous knowledge of the network were on focus of this work. The MCNC and related problems as well as possible solutions for it are the basis of several works [10] [16] [3]

2 Related Work

2.1 Maximum Connected Network Cover - Avrachenkov et Al.

Avrachenkov et Al. [3] are analyzing sampling algorithms on social networks towards the Maximum Connected Network Cover (MCNC) problem. The MCNC problem can be described as maximizing the amount of nodes and neighboring nodes recruited through the sampling algorithm with as least effort as possible. They analyzed many common sampling algorithms like Random Walk, BFS, DFS, and the Greedy Oracle algorithms of Guha and Khuller [10]. They also provided an own approach for solving the MCNC problem, the Maximum Expected Uncovered Degree (MEUD) algorithm and an approximation for it, the Maximum Observed Degree (MOD) algorithm. The networks they used for sampling are mostly large social networks out of the Stanford Network Analysis Project (SNAP) library. In contrary to most of the research works on network cover, they assume the topology of the social network not to be known in advance, since this is often not the case in the field.

The conclusion of their work is that they could provide an algorithm (MEUD / MOD) that outperforms the common sampling algorithms on the MCNC problem, if they do not know the network topology in advance. But their work also raises the question why some of the sampling algorithms solve the MCNC problem significantly faster than other algorithms, which only differ in few aspects. The work of Avrachenkov et Al. was the basis of our choice which sampling algorithms we want to analyze and the networks we want to sample. We also wanted to get on the bottom of the question why some algorithms perform better than others.

2.2 Minimum Connected Dominating Set - Guha and Khuler

Guha and Khuller [10] work on solving the MCDS (Minimum Connected Dominating Set) problem. The MCDS problem aims to find a connected subset of nodes with the minimum cardinality such that all nodes in the Graph are either in the subset or are neighbors to a node in the subset. Therefore the problem is similar to the MCNC problem. The MCDS problem is known to be NP-hard and is also related to the traveling tourist problem. In their work Guha and Khuller present two algorithms to efficiently approximate a solution for the MCDS problem. The first algorithm is a naive greedy algorithm which grows a tree to find a connected dominating set. In this work the algorithm is reffered to as Greedy Oracle (GO) due to its abilitiy to look one step ahead. This name was introduced by Avrachenkov et Al. [3]. The second algorithm is a simple modification of the first algorithm which delivers a even better performance. It builds different sets which dominates as many nodes as possible and connects them to a single dominating set. But this algorithms needs knowledge of the whole graph and therefore is less comparable to common sample algorithms. Guha and Khuller also provide an efficient implementation of this algorithm.

2.3 Biased Sampling - Maiya and Berger-Wolf

Maiya and Berger Wolf [16] are focussing their work on biased sampling with different algorithms and the properties of biased samples. In contrary to other works, they do not focus on problems of biased sampling, or how it could be avoided. They show that certain biases can be benefitial for some applications and metrics. They analyzed different commonly used sampling algorithms on up to 12 networks. The networks are chosen to represent a big variety of possible networks, among them big social- (Slashdot) and p2p-networks (Gnutella31). The metrics they're analyzing are local and global clustering coefficient, different measures for network reaches, the degree distribution similarity and the so called hubs. The degree distribution similarity is a measure of how strong the degree distribution of the sample differs from the original degree distribution. The hubs metric showes the ability of the sampling algorithm to quickly select nodes with a significant high degree (hubs). Their analysis showed, that the BFS is one of the worst performing sampling algorithms in discovering the netowrk and acquiring well-connected hubs. Another conclusion of their work is that simply chosing nodes with a high number of connection to the sample is a good approximation for sampling high degree nodes. They also show how biases can be exploited with real world applications like marketing and disease outbreak detection.

2.4 Sampling based Network analysis - Tim Grube

The work of Tim Grube [9] focussed on the properties of network samples computed with a wide variety of sampling algorithms in comparison with the properties of the whole graph. In contrary to our work, he did not use dynamic samples, instead he analyzed samples of a fixed size. Tim Grube analyzed a wide range of sampling algorithms: Uniform Sampling, different Random Walk algorithms, different Random Stroll algorithms, Frontier Sampling, Random Jump, BFS, DFS, Snowball Sampling, Forest Fire, and Respondent Driven Sampling. For the analysis he used generated graphs with a fixed size of 10,000 nodes. The graphs were generated with different network models like Regular, Random (Erdös-Renyi), Scale-Free (Barabási-Albert), Small-World (Watts-Strogatz), Rich-Club (Zhou-Mondragon), Community based (Brandes), and Clique based. After generating the samples with the different sampling algorithms he analyzed them with many metrics and compared the values with those of the original graph.

One of the results of his work was the conclusion that simple sampling algorithms like BFS and DFS producing better and more consistent samples as complicated algorithms like Randwom Walk with Degree Correction and Respondent-Driven Sampling. This result leaded us to the decision to skip many complicated Random Walk and Random Stroll approaches. Another interesting result was the sighting that DFS is behaving like a Random Walk, for many metrics. His conclusion was that they behave similar, only differing in the dead end behavior. This directly contradicts the result of Avrachenkov et Al. [3], which aroused our attention, so we decided to deeper analyze this aspect.

Tim Grube was a big inspiration for our selection of sampling algorithms, metrics and generated networks. His implementation is based on the GTNA framework [21] which was also developed at the Technical University Darmstadt.

2.5 Further Works on sampling properties

Stutzbach et Al. [24] [23] are working on the sampling of large dynamic graphs and unstructured peer-to-peer networks. Their work focusses on biased sampling and the analysis of specific properties of a graph. The sampling algorithms they use are BFS, DFS and several random walk based algorithms, like random stroll, random walk with degree correction and the metropolized random walk. As a result Stutzbach et Al. show some possibilities of correcting biased sampling.

Ribeiro's work [19] [20] concentrates on random walks and the characteristics and biases of the samples generated with these algorithms. In [19] he proposes a new mutlidimensional random walk, which we refer to as Frontier Sampling (FS). We decided to also analyse the properties of this algorithm regarding the MCNC problem. In [20] the focus lies on directed graphs and the avoidance of biases. For this reason he introduced different random walk and random jump algorithms. We scanned these algorithms but based on the work of Tim Grube [9] we decided to not consider them in our analysis because we do not expect a large difference to the other random walk algorithms regarding the MCNC problem.

Leskovec [15] is analyzing samples of large graphs. He used and introduced many measures of network properties which we also adapted, like the degree distribution (DD), the clustering coefficient (CC) and the diameter. For his analysis he used several sampling algorithms like BFS, DFS, Uniform Sampling and Forest Fire. He also introduced the Random Jump sampling algorithm, which we decided not to use in our analysis. This decision is based on the fact that the algorithm needs informations which we consider not to be available if we explore (social) networks. On the other hand we do not expect a significant performance of the Random Jump algorithm regarding the MCNC problem.

Kurant et Al. analyzed the properties of BFS and BFS like sampling algorithms. For their analysis they used generated random graphs with 10,000 nodes and a heavy tailed degree distribution. The algorithms they analyzed were Snowball Sampling, Respondent-Driven Sampling, BFS, DFS and Forest Fire (which we refer as Forest Fire non Revisisting - FFnr). The work of Kurant et Al. were an inspiration for our selection of the sampling algorithms and we directly adapted their implementation.

The work of Krishnamurthy et Al. [13] are focussing on sampling internet networks. For their analysis they use BFS and DFS among other sampling algorithms. Krishnamurthy et Al. and Ribeiro [19] are evaluating the local clustering coefficient which we adapted.

The work of Kim, Beznosov, and Yoneki [11] concentrates on the selection of influential neighbors in a graph. Kim and Yoneki formulated the Influental Neighbor Selection (INS) problem [12] which aims to find the most influental neighbor of a node to effectively spread information over the whole graph. They analyzed four different selection strategies, based on degree, propagation-weight, random selection, or a hybrid of propagation-weight and degree. These strategies were a inspiration for our start node selection strategies.

3 Preliminaries

In this Chapter, we define the commonly used terms in this thesis. First of all we explain the general terms like graph, neighborhood, degree, sample and batch. Then we go into detail on sampling algorithms, the sampling progress and resulting of this the progress zones, the progress neighborhood and the progress degree.

Graphs

A graph is defined as G := (V, E), with a set of nodes V and a set of edges E.

$$\begin{aligned} G &:= (V, E) \\ V &:= (v_1, v_2, v_3, ..., v_n) \\ E &\subset (V \times V) \end{aligned}$$

Since we only use undirected graphs, we don't have to distinguish between directed and undirected graphs. An edge is represented as an unordered pair of nodes $\{v_i, v_j\}$, with $i, j \in (0, ..., n)$. For simplicity we write $(v_i, v_j) = \{v_i, v_j\}$ from now on, which implies $(v_i, v_j) \equiv (v_j, v_i)$. In general the edge (v_i, v_j) can only exist once. We do not allow self loops, therefor no edge (v_i, v_i) can exist. The amount of nodes and edges in a graph are indicated with n and m.

$$\begin{aligned} n &:= |V| \\ m &:= |E| \end{aligned}$$

Neighborhood

N(v) defines the neighborhood of a node v.

$$N(v) := \{w \in V : (v, w) \in E\}$$

We also define the neighborhood for a set of nodes $M \subset V$.

$$N(M) := \{w \in V \setminus M : \exists v \in M : (v, w) \in E\}$$

Degree

Degree refers to the amount of edges a node is connected with. The degree for a graph is defined as k_v.

$$k_{v_i} := |\{(v_i, v_j) \in E : v_j \in V\}| = |N(v_i)|$$

Sample

A sample of graph $G := (V, E)$ is a subgraph $G_s(t) := (V_s(t)\,,\ E_s(t))$ of G at the timestamp t.

$$\begin{aligned} G_s(t) &:= (V_s(t)\,,\ E_s(t)) \\ V_s(t) &\subseteq V \\ E_s(t) &\subseteq E \end{aligned}$$

Batch

A batch is an set of node additions and edge additions $B(t) = (V^+(t), E^+(t))$. A batch B(t+1) represents the nodes and edges that are added from graph G to the sample $G_s(t)$. $V^+(t)$ is the set of nodes that are added to the sample while $E^+(t)$ is the set of edges between the nodes that are added and between the nodes that are added and those nodes that are already in the sample.

$$\begin{aligned} B(t) &:= (V^+(t)\,,\ E^+(t)) \\ V^+(t) &\subset V \setminus V_s(t) \\ E^+(t) &:= \{(v,w) \in E : v \in V^+(t) \ \wedge \ w \in V_s(t)\} \subset E \setminus E_s(t) \\ G_s(t+1) &:= (V_s(t+1)\,,\ E_s(t+1)) \\ V_s(t+1) &:= V_s(t) \cup V^+(t+1) \\ E_s(t+1) &:= E_s(t) \cup E^+(t+1) \end{aligned}$$

We define the batchsize |B(t)| as the amount of nodes added with the batch.

$$|B(t)| := |V^+(t)|$$

Sampling Algorithm

A sampling algorithm is an algorithm which samples a given graph G step by step. We define a sampling algorithm $A(G, G_s(t))$ as a generator for batches, which will be added to the sample $G_s(t)$. In the beginning the sample $G_s(0)$ is empty and then the algorithm will create a batch in every step which is added to the sample.

$$\begin{aligned} G_s(0) &:= (\emptyset, \emptyset) \\ A(G, G_s(t)) &= V^+(t+1) \\ E^+(t+1) &:= (v_i, v_j) \in E : v_i \in V^+(t+1) \ \vee \ v_j \in V^+(t+1) \end{aligned}$$

In every step t, the algorithm performs the amount of bs visitation steps. The current visitation step is reffered to as $z \in (1...bs)$. In every visitation step the algorithm visits a node v_z and adds it to the batch.

Sampling progress

While in the sampling progress (A operates on graph G), nodes in the graph have one of three states. A node is either visited, seen, or unseen.

$$State(v,t) := \begin{cases} Visited, & \text{if } v \in V_s(t) \\ Seen, & \text{if } v \in N(V_s(t)) \ \wedge \ v \notin V_s(t) \\ Unseen, & \text{else} \end{cases} \tag{3.1}$$

Visited: A node that was visited by the sampling algorithm and is already added to the sample. All these nodes were once selected as v_z from the sampling algorithm.

Seen: A node that has not yet been visited by the sampling algorithm, therefore it is not in the sample but it is in the neighborhood of the nodes in the sample. Every node we visit has a list with its neighbors and therefore we know the neighbors of the nodes which we have already visited.

Unseen: A node that has not yet been visited and it is not in the neighborhood of the sample. The sampling algorithm is unaware of this node.

Remark: Per definition, a node that is visited, is not seen anymore!

Progress zones

As we now have introduced states for the nodes, we can define progress zones of the sampling algorithm $A(G, G_s(t))$. The zone with the visited nodes is called $\mathbb{V}(t)$.

$$\mathbb{V}(t) := \{v \in V_s(t)\} = \{v \in V : State(v,t) = Visited\} \tag{3.2}$$

The zone with the seen nodes is defined as the neighborhood of $\mathbb{V}(t)$, $\mathrm{N}(\mathbb{V}(t))$. This zone is is called $\mathbb{S}$(t). Per definition it is the same as:

$$\mathbb{S}(t) := N(\mathbb{V}(t)) = \{v \in V : State(v,t) = Seen\} \tag{3.3}$$

The zone with the unseen nodes is defined as $\mathbb{U}$(t) (3.4). Its complement $\bar{\mathbb{U}}(t)$ are all visited and seen nodes (3.5).

$$\mathbb{U}(t) := \{v \in V \setminus V_s(t)\} = \{v \in V : State(v,t) = Unseen\} \tag{3.4}$$

$$\bar{\mathbb{U}}(t) := \mathbb{S}(t) \cup \mathbb{V}(t) \tag{3.5}$$

Progress Neighborhoods

As we now have introduced states and zones for the progress, we can define some special types of neighborhoods. *Unvisited neighborhood*: The unvisited neighborhood is the neighborhood of a node (3.6) or a set of nodes $M \subset V$ (3.7), containing all neighbors which are not visited.

$$UN(v,t) := \{w \in V : (v,w) \in E \ \wedge \ w \notin V_s(t)\} \tag{3.6}$$
$$UN(M,t) := \{w \in V \setminus M : \exists v \in M : (v,w) \in E \ \wedge \ w \notin V_s(t)\} \tag{3.7}$$

Visited neighborhood: The visited neighborhood is the neighborhood of a node (3.8) or a set of nodes $M \subset V$ (3.9) containing all neighbors which are visited.

$$VN(v,t) := \{w \in V_s(t) : (v,w) \in E\} \tag{3.8}$$
$$VN(M,t) := \{w \in V_s(t) : \exists v \in M : (v,w) \in E\} \tag{3.9}$$

Progress Degree

With the progress zones we can also define degrees for nodes, depending on the progress. The observed degree $k^o_{v_i}$ is the amount of neighbors that are already visited.

$$k^o_{v_i} := |\{v_j \in \mathbb{V}(t) : (v_i, v_j) \in E\}| \tag{3.10}$$

The uncovered degree $k^u_{v_i}$ is the amount of neighbors that are unseen.

$$k^u_{v_i} := |\{v_j \in \mathbb{U}(t) : (v_i, v_j) \in E\}| \tag{3.11}$$

4 Metrics

In this Chapter we describe the different metrics and values that we measure on the sample during the sampling process. The metrics are evaluated in each sampling step. The analysis of these metrics shows us in which way the different sampling algorithms biases the sample in comparison to the original graph.
In Table 4.1 we give an overview about abbreviations which we use for the metrics, before we describe the different metrics in detail.

Table 4.1: Metric abbreviation

Metric name	Abbreviation
Extent	EX
Degree Distribution	DD
Clustering Coefficient	CC
Betweenness Centrality	BC
Sampling Modularity	SM
All Pairs Shortest Path	APSP
Single Source Shortest Path	SSSP
Motifs	Motif

4.1 Extent

The Extent metric measures how far the sampling algorithm has progressed through the graph. For this reason the metric counts the amount of nodes in the different progress zones: The visited zone 3.2, the seen zone 3.3, and the unseen zone 3.4.
The metric computes four different values:

$$\begin{aligned} Visited\_Nodes &:= |\mathbb{V}(t)| \\ Seen\_Nodes &:= |\mathbb{S}(t)| \\ Unseen\_Nodes &:= |\mathbb{U}(t)| \\ Seen\_and\_Visited\_Nodes &:= |\bar{\mathbb{U}}(t)| \end{aligned}$$

4.2 Degree Distribution

The degree distribution is the probability distribution of the degree k of nodes $v \in V$ in a graph G. It is an indicator of the density of the graph.

$$P(x = d) := \frac{|\{v \in V : d(v) = d\}|}{|V|}$$

4.3 Clustering Coefficient

The clustering coefficient is a measure for the connectivity in a network and how much nodes tend to cluster together. We have to consider three different coefficients. The local clustering coefficient C_{v_i} quantifies how close a node and his neighbors are to being a clique [25]. The average clustering coefficient C' is the average of the local clustering coefficient of all nodes [25] [19] [13]. The global clustering coefficient (also called transitivity) is a measure clustering of the whole network [6] [9].
The values are computed as follows:

$$\begin{aligned} C_{v_i} &:= \frac{number\ of\ closed\ triplets\ connected\ with\ v_i}{number\ of\ connected\ triplets\ centered\ at\ v_i} \\ C' &:= \frac{1}{n}\sum_{i=0}^{n} C_{v_i} \\ transitivity &:= \frac{number\ of\ closed\ triplets}{number\ of\ connected\ triplets} \end{aligned}$$

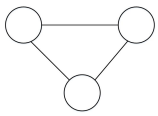

Figure 4.1: Closed triplets

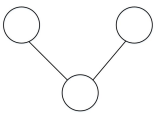

Figure 4.2: Connected triplets

4.4 Betweenness Centrality

The betweenness centrality measures the centrality of a node in a network. The metric was developed by Anthonisse and Freeman [2] [7] and counts the amount of shortest paths in the network over this node. The more shortest paths going through this node, the more network traffic the node has to handle.

4.5 Sampling Modularity

Sampling Modularity indicates the fraction between the sampled and the unsampled part of the network. We use two different types of the sampling modularity. Sampling Modularity V1 measures the amount of edges in the sample and divides them through the amount of edges in the whole graph. The Sampling Modularity V2 measures the count of edges between the sample and the rest of the network |F(t)| and again divides the amount of edges in the sample $|E_s(t)|$ through this value |F(t)|. The Sampling Modularity V2 metric is inspired by Newman [18] [9].

$$
\begin{aligned}
SamplingModularityV1 &:= \frac{|E_s(t)|}{|E|} \\
F(t) &:= \{(v_i, v_j) \in E : v_i \in \mathbb{V}(t) \wedge v_j \in \mathbb{S}(t)\} \\
SamplingModularityV2 &:= \frac{|E_s(t)|}{|F(t)|}
\end{aligned}
$$

4.6 All Pairs Shortest Path

This metric computes the shortest path from one node to another node, for all possible pairs in the whole graph. With the paths between all nodes we can compute four different values: The diameter, the characteristic path length, the existing paths, and the possible paths. The diameter is the highest shortest path length between two components.

$$
Diameter := max(|shortestPath(v_i, v_j)|)
$$

The characteristic path length is a measure for the average shortest path length in a graph. It is computed as follows:

$$
Characteristic\ Path\ Length := \frac{\sum_{v_i, v_j \in V} |shortestPath(v_i, v_j)|}{n * (n-1)}
$$

The existing paths value is the count of existing paths in the graph. The possible paths value is the amount of paths that could be possible in the graph.

$$
Possible\ Paths: := n * (n-1)
$$

4.7 Single Source Shortest Path

This metric computes the shortest path from one single node to all other nodes in the graph. While APSP computes this path for all node combinations, SSSP computes this only for a single node. Computed on more than one node, SSSP can be used to approximate the values of diameter, existing paths, and characteristic path length. The possible paths value can be measured exactly. In cases of graphs with more than 20.000 nodes, the computation of the SSSP for just 1% of the nodes results in a very good approximation of these values.

4.8 Motifs

Motifs are simple building blocks, constellations of nodes and edges, but the amount and structure of these building blocks differ between networks based on the network purpose. R. Milo et

Al. [17] have written a highly recognised paper about these motifs and how their amount can help classifying networks. Our interests in these motifs is that we want to know if it is possible to characterize a sampling algorithm based on amount of motifs that we found in the sample. The amount of motifs captured with different sampling algorithms can allow conclusions on the characteristic properties of the sampling algorithm. For this reasons we evaluated the motif metric. Our motif metric simply counts how often a specific motif exists in the sample. The examined motifs are listed below.

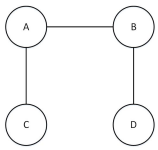

Figure 4.3: Motif 1

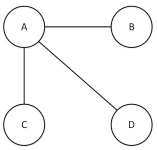

Figure 4.4: Motif 2

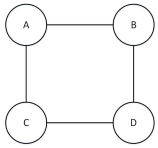

Figure 4.5: Motif 3

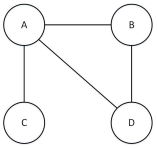

Figure 4.6: Motif 4

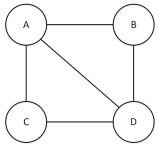

Figure 4.7: Motif 5

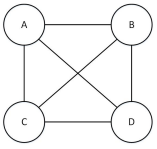

Figure 4.8: Motif 6

5 Sampling Algorithms

5.1 General Properties

In general a sampling algorithm is an algorithm that travels through a graph and visits nodes based on its specific implementation. In our case a sampling algorithm works as described in 3.1. The sampling algorithm works as a generator for batches, which are added to the sample. The algorithms differ in the way they are selecting the nodes that are added with the batch.

If we look at sampling algorithms we have to discuss the term of revisiting. Some sampling algorithms allow the visitation of nodes that were already visited (and therefore these ndoes are already added to the batch or sample). Sampling algorithms that do not allow revisiting and thus are aware of the visitation status are called self-aware. Since our metrics do not evolve with revisited nodes and revisited nodes have a negative impact on the progress for the MCNC focused algorithms, we only considered algorithms that do not allow revisiting. We also took a look are algorithms that are not self-aware, but if we used them for our analysis, we implemented nr (= non revisiting) versions of these algorithms.

Another important point for us was the selection of the first node. Every sampling algorithm begins its sampling process with a single node. In most of the related work to this topic, this start node is chosen randomly. We decided to also analyze the influence of the start node selection on the performance of the sampling algorithms. Therefore we introduce some start node selection strategies in Chapter 5.4.

Table 5.1: Sampling algorithm abbreviation

Sampling algorithm name	Abbreviation
Random Walk	RW
Random Walk non Revisiting	RWnr
Frontier Sampling	FS
Depth First Sampling	DFS
Randomized Depth First Sampling	DFS_random
Breadth First Sampling	BFS
Forest Fire	FF
Forest Fire non Revisiting	FFnr
Respondent Driven Sampling	RDS
Snowball Sampling	SS
Maximum Observed Degree	MOD
Maximum Expected Uncovered Degree	MEUD
Greedy Oracle	GO
Uniform Sampling	US

5.2 Algorithms

5.2.1 Random Walks

Random Walk

The Random walk is a sampling algorithm which randomly chooses the node it will visit next out of the neighbors of the currently visited node. In particular this means, in visitation step z+1 RW randomly chooses a neighbor of the node v_z visited in step z and visits it. In this setup it could be the case that RW chooses a node that was already visited before. Considering this issue you have to distinguish between two kinds of random walk algorithms: The random walk that allows revisiting (RW) and the random walk without revisiting (RWnr). The random walk that allows revisiting does not differ between nodes it has already visited and those it has not, it will just proceed. [23] [15] [19] [3]

Random Walk without Revisiting

The Random walk without revisiting (RWnr = Random Walk non revisiting) is a modification of the classic random walk. In step z+1 RWnr randomly choses a neighbor out of the unvisited neighbors of v_z, $UN(v_z)$. In the case that v_z does not have any unvisited neighbors ($|UN(v_z)| = 0$), RWnr will randomly choose a node v_i out of $\mathbb{V}$(t). Now RWnr will select a random node out of $UN(v_i)$. If this again isn't possible it will proceed with chosing a new v_i out of $\mathbb{V}$(t). If no node in $\mathbb{V}$(t) have any unvisited neighbor left, the algorithm will stop.

Frontier Sampling

Frontier Sampling is a Random Walk without revisiting with multiple walkers. This means we have a pool of l nodes from which we can proceed our Random Walk. Frontier Sampling always picks the node v_z with the highest degree k out of the l nodes in the pool. Our implementation of Frontier Sampling is inspired by Ribeiro [19]. For l = 1, Frontier Sampling behaves exactly like the RWnr algorithm.

5.2.2 DFS based Algorithms

Depth First Sampling

Depth-First Sampling is working like the Depth-First Search algorithm. It is queuing all unvisited neighbors of the currently visited node in a LIFO-queue. In the next step, it picks the next node in the queue and visits it. As a result, DFS explores the nodes that are faraway (in the number of hops) from the seed first [14].

Randomized Depth First Sampling

The randomized DFS algorithm is a slightly modificated DFS. The framework we are using for our implementation has the property to access neighbors from a node in a somehow ordered way. To eliminate the influence of these order, we developed the randomized DFS. It simply mixes the order in which the neighbors of the current node are added to the LIFO queue. In all other concerns it is identical to the DFS.

5.2.3 BFS based Algorithms

Breadth First Sampling

Breadth First Sampling is working like the Breadth First Search. It is a classic graph traversal algorithm that starts from the seed and progressively explores all neighbors. It is queuing all unvisited neighbors of the currently visited node in a FIFO-queue. In the next step, it picks the next node in the queue and visits it. Thus, BFS discovers all nodes within some distance from the seed [14].

Forest Fire

Forest Fire is a modified version of the BFS. The difference is that not all neighbors of the node v_z in visitation step z are visited. It flips a coin for every neighbor $v \in N(v_z)$ and with probability of success p, it decides to put v in its FIFO queue. In visitation step z+1 it picks a node out of the FIFO queue and visits it. Since v is picked out of all neighbors of v_z, FF can revisit nodes.

Forest Fire without Revisiting

FFnr is the non revisiting version of the Forest Fire algorithm. The difference is that FFnr picks v out of $UN(v_z)$, the not yet visited neighbors. Therefore FFnr cannot visit nodes that were visited before. FFnr reduces to BFS for p=1 [14]. Our implementation of FFnr is inspired by Kurant et Al. [14].

Respondent Driven Sampling

RDS is a sampling algorithm and modification of the BFS algorithm. At each visitation step z, RDS randomly selects l neighbors out of all neighbors of the current node $N(v_z)$ and schedules them into a FIFO queue. RDS then picks the next node in the queue and visits it. Our implementation of RDS was inspired by Kurant et Al. [14]. But since RDS allows revisiting, we decided to not use it for our analysis. RDS with l=1 behaves like a RW.

Snowball Sampling

Snowball Sampling is a modification of the BFS algorithm and more specific the non revisiting version of RDS. According to a classic definition by Goodman [8], an l-name Snowball Sampling

is similar to BFS, but at every node v_z, not all unvisited neighbors in $UN(v_z)$, but exactly l neighbors are chosen randomly. These l neighbors are scheduled to visit in a FIFO queue. Our implementation of SDS was inspired by Kurant et Al. [14]. Snowball Sampling with a high l value (l > degree of all nodes) behaves like a BFS while SDS with a l value of 1, behaves like a RWnr.

5.2.4 Maximum Coverage Algorithms

These algorithms are designed to reach full network coverage (all nodes are either seen or visited) with as few steps as possible.

Maximum Observed Degree

This sampling algorithm greedily chooses the next node to visit based on the highest observed degree k_v^o. In particular the algorithm chooses the node in $\mathbb{S}(t)$ with the most connections to already visited nodes (nodes in $\mathbb{V}(t)$). This algorithm was developed under the assumption that we do not know which degree unvisited nodes have. The only thing we know (because we can compute this value while processing the algorithm) is the count of connections to already visited nodes. The assumption is that the neighbor with the highest observed degree also has a high degree at all. MOD was introduced by Avrachenkov et Al. [3] and it is an approximation for MEUD.

Maximum Expected Uncovered Degree

MEUD is a sampling algorithm developed by Avrachenkov et Al. [3], which chooses the next node to be visited based on the observed degree (3.10) and the degree distribution. With the observed degree and the degree distribution the algorithm computes an assumption of the degree of a not yet visited node. MEUD computes this assumptions for all nodes in $\mathbb{S}(t)$ and then chooses the node with the highest assumed degree as the next node to be visited.

Greedy Oracle

Greedy Oracle is a sampling algorithm which chooses the next node to be visited based on the uncovered degree of the nodes. In the visitation step z, GO chooses the node in $\mathbb{S}(t)$ with the highest uncovered degree $k_{v_z}^u$. The algorithm needs to know the edges of all neighbors of the nodes in $\mathbb{S}(t)$ to function correctly. This means the algorithm needs a one step lookahead for neighbors. The Greedy Oracle algorithm was developed by Guha and Khuller [10].

5.2.5 Other

Uniform Sampling

Uniform Sampling is a sampling algorithm that randomly chooses the next node from all nodes in V and visits it. A typical approach is selecting random node ids but this is barely efficient if

the id-space is not well populated or unknown [23] [19] [15]. To avoid this problem, we always used well populated id spaces. Additionally our implementation does not allow revisiting, so we randomly select nodes out of all unvisited nodes.

5.3 Expectations

In this Section we describe how we think the properties of the samples, produced by the sampling algorithms, will look like. We describe these properties on the basis of the metrics introduced in Chapter 4. We divided the sampling algorithms into five groups based on their behavior.

5.3.1 Random Walks

This group includes the Random Walk with and without revisiting, Frontier Sampling, Respondent Driven Sampling with l=1 and Snowball Sampling with l=1.

Extent

We assume that Random Walks will explore the network pretty fast and reach high *Seen_Nodes* values in the beginning of the sampling process. But the Random Walk based sampling algorithms will have problems to find nodes that have not been seen or visited in the end which we think will result in strongly decreasing *Seen_Nodes* value after a time and in the same time a only slowly growing *Seen_and_visited_Nodes* value.

Degree Distribution

We expect that the Degree Distribution will converge towards the original distribution of the network fastly.

Clustering Coefficient

Again we expect that the sample produced by Random Walk based sampling algorithms will pretty fast have the same average and global Clustering Coefficient than the original network. Because of the randomly chosen nodes we expect a broad variety of different nodes, which should characterize the whole network.

Sampling Modularity

The Sampling Modularity V1 will congerge to one. We expect that the value will converge faster in the beginning and more slowly in the end. The trend should be comparable to the Extent *Seen_and_Visited_Nodes* value but less significant. We expect Sampling Modularity V2 value will be slowly growing over the time and have an exponential-like growth in the end because only few nodes are not sampled yet and the amount of edges towards them is low.

All Pairs Shortest Paths

We expect the Diameter and the Characteristic Path Length of the sample to reach their original values very fast. As we do not assume any characteristic trends, the Random Walk might be a good baseline for this values.

Motifs

As for the Diameter and the CPL we do not expect any characteristic trends here and therefore the RW might be a good baseline for the Motifs metric.

5.3.2 DFS based Algorithms

This group only contains the classical Depth First Sampling algorithm.

Extent

Especially regarding the results of Avrachenkov et Al. [3] the DFS have a very poor performance regarding the MCNC problem which leads us to the expectation that the DFS will have a very slow growing *Seen_and_visited_Nodes* value.

Degree Distribution

We expect that the Degree Distribution will converge towards the original distribution of the network.

Clustering Coefficient

We expect that DFS will start with a very low average Clustering Coefficient that will converges to the original value of the network. The global Clustering Coefficient is strongly related to the structure of the graph, but we assume a very biased value in the beginning too.

Sampling Modularity

The Sampling Modularity V1 will converge to one. We expect the value to converge more slowly than the value of RW because we assume DFS will sample less edges. The trend should be comparable to the Extent *Seen_and_Visited_Nodes* value but less significant. We expect Sampling Modularity V2 value will have the same trend like the RW will show, but DFS should have a significant lower value over the time, since it should have pretty many edges between the sample and the unvisited part of the graph.

All Pairs Shortest Paths

We expect the Diameter and the Characteristic Path Length the be extremely biased. These measures will both start with very high values and then slowly converge to the original value. We assume a very significant difference to every other sampling algorithms in this metric.

Motifs

We expect this metric to be strongly biased if the sample is generated with a DFS algorithm. Especially we expect that DFS will have a trend towards motifs 1 and 2. We assume to see a significant difference between DFS and other sampling algorithms.

5.3.3 BFS based Algorithms

This group includes the Breadth First Sampling, Forest Fire with and without Revisiting, Respondent-Driven Sampling and Snowball Sampling.

Extent

As Avrachenkov et Al. [3] shows, BFS has a better performance than DFS regarding the MCNC problem. This leads us to the expectation that BFS will show the same behavior as the Random Walk based sampling algorithms but with a slighly lower *Seen_and_Visited_Nodes* value.

Degree Distribution

We expect that the Degree Distribution will converge towards the original distribution of the network fastly.

Clustering Coefficient

The Breadth First Sampling should directly converge to the original value. We expect the BFS to be a good algorithm to fastly measure the average Clustering Coefficient.

Sampling Modularity

For the Sampling Modularity value V1, BFS should produce the same or only slightly different results as the Random Walk based sampling algorithms. Sampling Modularity V2 instead should show an strong exponential growth in the end.

All Pairs Shortest Paths

We expect the Diameter and the Characteristic Path Length to start with very low values which slowly increase and converge to the original value.

Motifs

The Motifs metric should not show any significant trend if the sample is generated with a BFS like sampling algorithm.

5.3.4 Maximum Coverage Algorithms

The group of Maximum Coverage Algorithms contains the Maximum Observed Degree and the Greedy Oracle sampling algorithm. The Maximum Expected Uncovered Degree sampling algorithm is not part of our expectations because we cannot predict its behavior since it was not implemented by anyone yet.

Extent

In most of the cases, samples generated by this group of algorithms should show a faster increasing *Seen_and_Visited_Nodes* value than samples of other sampling algorithms.

Degree Distribution

We expect the Degree Distribution to be biased towards nodes with a higher degree. Since these sampling algorithms concentrate to reach as many nodes as possible, they have a tendence to visit nodes with a higher degree more likely.

Clustering Coefficient

We do not expect a special trend here. The Maximum Coverage sampling algorithms should directly converge to the original value. We expect them to be good algorithms to fastly measure the average Clustering Coefficient.

Sampling Modularity

Especially MOD should have a higher Sampling Modularity V1 and V2 value than other sampling algorithms, since MOD always picks the next node with the highest number of edges to the nodes in the sample. This should result in many edges in the sample and therefore in a high Sampling Modularity V1 and V2 value. GO on the other hand should show a rather low SM V2 value because it picks nodes based on the amount of net yet seen neighbors which should result in many edges between nodes in the sample and nodes that have not been sampled yet.

All Pairs Shortest Paths

We do not think that MOD and GO should show any significant trend in Characteristic Path Length and Diameter measures.

Motifs

The Motifs metric should not show any significant trend if the sample is generated with MOD or GO.

5.3.5 Uniform Sampling

We treat Uniform Sampling as an own group because it does not fit into the other groups.

Extent

Our expectation towards the Uniform Sampling is that it will have a poor performance towards the MCNC problem and therefore it will show a slowly growing *Seen_and_Visited_Nodes* value. Anyway we expect it to be better than the DFS.

Degree Distribution

We expect that the Degree Distribution will converge towards the original distribution of the network fastly.

Clustering Coefficient

US should show the lowest average Clustering Coefficient over the sampling process of all sampling algorithms. It will slowly converge from a very low value to the original average CC of the graph. The global Clustering Coefficient should converge to the original value very fast and US might be a good baseline for this metric value.

Sampling Modularity

The Sampling Modularity values V1 and V2 should show the same trends like those measured with the DFS algorithm.

All Pairs Shortest Paths

We expect the Diameter and the Characteristic Path Length of the sample to reach their original values very fast.

Motifs

The values of the Motifs metric should be very unbiased. We assume that Unifom Sampling might be a good baseline for this metric.

5.4 Start Node Selection

A start node selection strategy is a simple algorithm that chooses one single node v out of V. The chosen node v will be the first node in the batch with z=1 and t=0. The sampling algorithm will start the sampling progress based on this start node. Every start node selection strategy can be combined with every sampling algorithm.

Table 5.2: Start node selection strategies

Start node selection strategy name	Abbreviation
ID Selection	IS
Random Selection	RS
Highest Degree Selection	HDS
Highest Random Degree Selection	HRDS
Highest Random Degree Sum Selection	HRDSS
Node Value List Selection	NVLS

5.4.1 ID Selection

The ID Selection is the most simple start note selection strategy. It picks the start node with a specific pre defined ID. This can simulate runs where only one specific node from the whole network is known in advance, or to test the influence of a very well connected node.

5.4.2 Random Selection

Random Selection is the standard selection strategy. It randomly picks one node v_i out of all nodes V. We use this strategy in all runs where we do not explicitly want to test the influence of the start node.

5.4.3 Property based Selections

Highest Degree Selection

The highest Degree Selection selects the node v_i with the highest degree k_{v_i} in the whole graph G. We introduce this selection strategy to test the influence of background knowledge especially regarding the MCNC problem.

Highest Random Degree Selection

The Highest Random Degree Selection of grade l randomly picks the amount of l nodes from V. It then chooses the node with the highest degree to be the start node. This simulates a situation where we can access at least l nodes from a network but do not have further background knowledge. The strategy aims to find the most influential node in our range.

Highest Random Degree Sum Selection

The Highest Random Degree Sum Selection of dimension l, o randomly picks a set of l nodes from V. From every v_i in l, it randomly picks a set of o nodes from $N(v_i)$, so we have o neighboring nodes of v_i. It sums up the degree of v_i and the o neighboring nodes of v_i. This value is called the degree sum. In the end the strategy chooses the node v in l with the highest degree sum. This selection strategy simulates a situation where we can access at least l nodes and o of their neighbors but do not have further background knowledge. It aims to find the most influential cluster or neighborhood in our range.

Node Value List Selection

The Node Value List Selection is a very generic start node selection strategy. It receives a node value list, a list where all nodes from the graph are listed with a specific value. It then picks either the node with the highest, lowest or median value, as it was specified. As node value we can pick the local clustering coefficient or the degree for example. This start node selection strategy therefore simulates perfect background knowledge.

6 Implementation

In this Chapter we will introduce our implementation and the technical background. For our analysis we use the Dynamic Network Analyzer [22] a java framework which was developed at the Technical University of Darmstadt. The framework was build to analyze the properties of dynamic generated networks. It supports undirected and directed graphs just as weighted and unweighted nodes and edges.

The core of the framework are graph and batch generators. Graph generators create whole graphs in a single step. Based on which graph generator you choose you can create completely random graphs or graphs build on specific laws or techniques, like Scale-Free or Small World networks. You can also read graphs from files in the native DNA format. Batch generators instead generate updates (node additions, node removals, edge additions, edge removals, changes of the node / edge weight) and combine them to batches of a specific size. These batch generators can generate updates randomly or based on rules and techniques.

The main purpose of the framework are so called runs. In a run a graph generator generates a graph and then in each step of the run a batch generator generates a batch which is added to the graph. These steps are similar to the steps t which we defined in Chapter 3. In every step several metrics can be measured on the graph. These metrics can be of different types, like values, distributions, or node value lists. Metrics can be evaluated before or after each update or before or after each batch. Furthermore the metrics can be recomputed completely in each step or just be updated if this is possible (relies on the type of metric).

The user can choose to create only a single run or a set of runs. If you choose to create more then one run, the data of the runs will be aggregated. After this process is finished, the data can be converted into gnuplot files for plotting.

The DNA framework has a rather good functionality to dynamically analyze graphs. We decided to use this funcionality to dynmaically analyze samples of sampling algorithms. For this reason we extended the DNA framework and added new functionality. The main extension is a new type of batch generator, the sampling algorithm. These class is a generic sampling algorithm. The sampling algorithm gets a set of parameters like the graph that it shall sample, the batchsize $|B(t)|$ and a start node selection strategy. Furthermore a limit of steps can be given to the sampling algorithm if only a fixed amount of steps shall be sampled. The concrete sampling algorithms like BFS, DFS etc. inherit from the abstract base class sampling algorithm.

The structure of our implementation can be seen in Figure 6.1.

The Start Node Strategies are implemented with an interface which allows to easily create new strategies, as shown in Figure 6.2. The concrete strategies, like Random Selection or ID Selection implement these interface.

Additionally we implemented three new metrics: The Extent metric (recomputation version, update is not necessary), the Sampling Modularity metric (recomputation and update version) and the SSSP metric (recomputation version). The SSSP metric gets gets the value l as parameter which indicates the number of nodes used to evaluate the metric. Due to their purpose the Extent and Sampling Modularity metrics only work in combination with sampling algorithms.

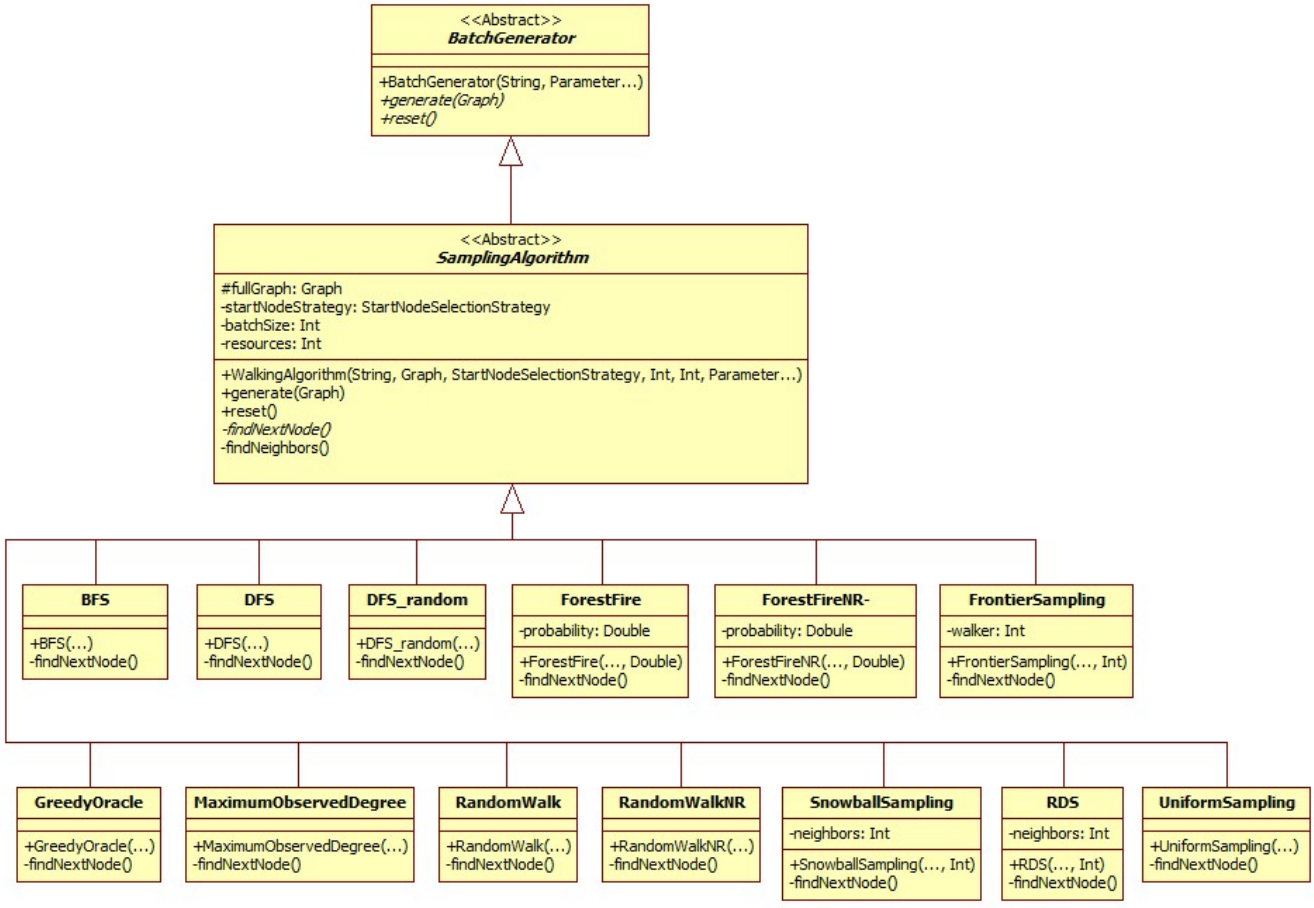

Figure 6.1: UML of Sampling Algorithm

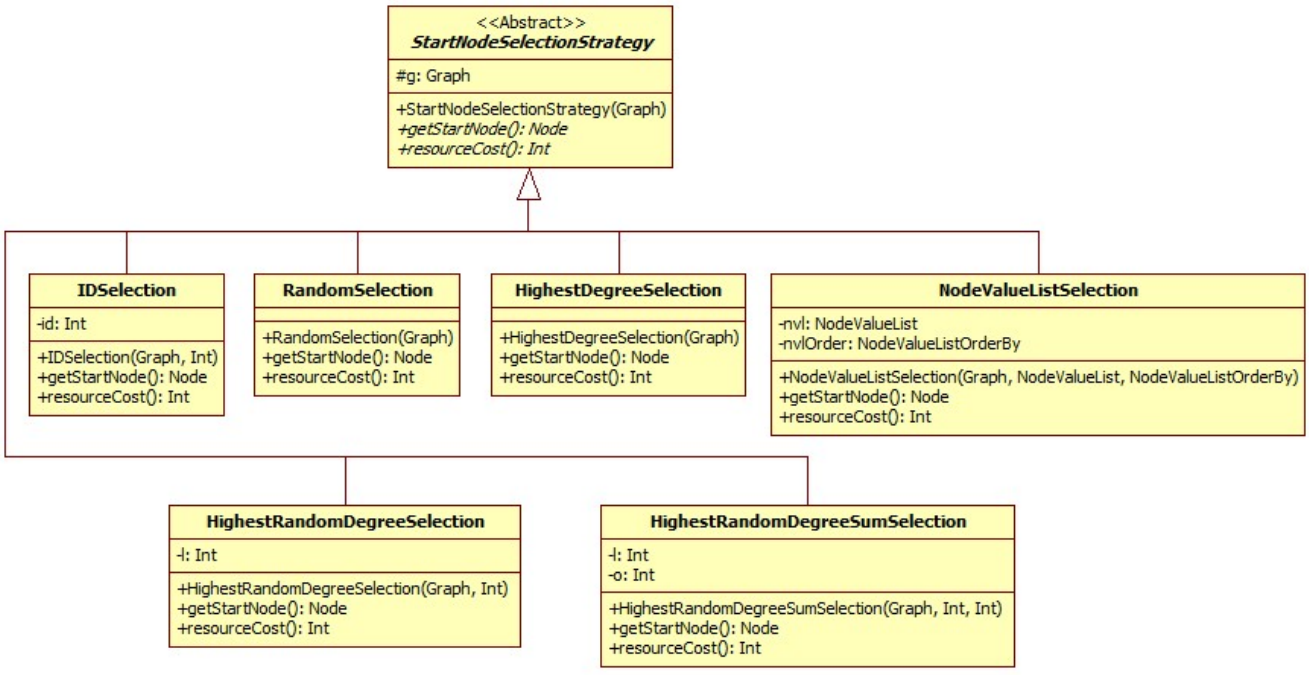

Figure 6.2: UML of Start Node Selection Strategies

All other metrics that we wanted to analyse, like the Degree Distribution, the Clustering Coefficient, the Betweenness Centrality, and the All Pairs Shortest Path were already implemented.

Another extension was a new graph reader, the SNAP graph reader. This reader was implemented to read graphs in the Stanford Network Analysis Project (SNAP) format and load them into the framework.

We created two executable jar files to easily run our extension of the framework. Both jar files need a data directory were the data can be read and written and a config directory with all the necessary configuration files.

The first executable is the PreProcessing.jar, which is used to preprocess the graphs which we want to use for our analysis. The preprocessing of the files consists of up to three steps, based on which options we chose. First of all we can transform graph files in the SNAP file format into the DNA graph file format. The second step is to delete self loops and select the biggest connected component in the graph. The non connected parts of the graph are cut off. In the third step we can compute several different metrics on the graph, which deliver a NodeValueList. These lists, together with the graph file are the result of the preprocessing, if we chose this option. The NodeValueLists can be used for the NodeValueList based start node selection. The available metrics at the moment are the local clustering coefficient and the betweeness centrality. The different options and steps that we can compute in the preprocessing are passed by parameters to the executable jar file. The different parameters and defaults are shown in Table 6.1. All parameters that are not strictly denoted as optional are mandatory.

Table 6.1: PreProcessing.jar Parameter

Position	Parameter name	Explanation
1	SNAP/DNA	Format of the graph file that we want to read
2	Filename	The filename of the graph file
3	Output Filename	The filename of the output graph file
4	connect/noconnect/undirected	If connect is selected, selfloops and non connected parts are cut off. If undirected is selected, directed graphs are treated as undirected graphs
5-6	LCC/BC	**OPTIONAL** Computes the LCC or CC nvl of the graph

EXAMPLE: java –jar PreProcessing.jar SNAP amazon.SNAP.txt amazon.DNA.txt connect lcc bc

This would read a graph in the SNAP format from the amazon.SNAP.txt file, select the biggest connected component, cut off self loops, and write this graph in the DNA format in the file amazon.DNA.txt. It would also compute the local Clustering Coefficient as well as the Betweeness Centrality of the graph, which would result in a amazon.DNA.txt.ClusteringCoefficient_0.txt and a amazon.DNA.txt.BetweenessCentrality_0.txt.

The sampling of the graphs is started with the second executable, the Sampling.jar. You can choose different parameter for the sampling runs, which are shown in Table 6.2.

Table 6.2: Sampling.jar Parameter

Position	Parameter name	Explanation
1	Name	Name of the series and the target directory
2	Filename	The filename of the graph file
3	Number of runs	The amount of runs that will be generated
4	Number of batches	The amount of batches per run
5	Batchsize	The amount of nodes added per batch |B(t)|
6	Number of first run	The number of the first run. Necessary if we want to proceed with already started runs
7	List of sampling algorithms	A list of algorithms in the form: Abbreviation, Parameter*, Abbreviation,...
8	Start Node Strategy	**OPTIONAL** with the keyword SNS3 you can choose a Start Node Selection Strategy with the format: SNS3, Abbreviation, Parameter 1, Parameter 2** **DEFAULT:** Random Selection
9	List of metrics	**OPTIONAL** A list of the metrics that shall be evaluated in the format: Metrics, Abbreviation, Abbreviation,... **DEFAULT:** Extent

* Only the sampling algorithms Forest Fire, Forest FireNR, Snowball Sampling, Respondent Driven Sampling and Frontier Sampling need additional parameters. All other sampling algorithms simply gets a zero as parameter here.

** Not all Start Node Selection Strategies need two parameters. If they need less than two parameters, just enter zero here.

EXAMPLE: java –jar Sampling.jar Series1 Graph.txt 5 100 20 0 GO 0 RW 0 FF 0.6 SNS3 hrdss 5 5 Metrics ddU ccR exR

This would start a series consisting of 20 runs, beginning with run 0. Every run consists of 100 batches and every batch of 20 nodes. The sampling algorithms Greedy Oracle, Random Walk and Forest Fire (with 0.7 propabilty) will sample the graph which is read from the Graph.txt file. The sampling algorithms would start with the “Highest Random Degree Sum Selection” start node selection strategy, which will select 5 nodes and their 5 neighbors to choose the group with the highest degree sum. The dynamic sampled graphs will be analyzed by the metrics Degree Distribution, which will be computed based on updates, the Clustering Coefficient and Extent which will be recomputed in each step. This series of runs will be called “Series1”.

7 Analysis

7.1 Networks

In this Section we introduce the different networks on which we analyzed the sampling algorithms. The selection was inspired by Avrachenkov et Al. [3], but we expanded the selection with generated networks. In Table 7.1 we give a general overview about the graphs. The different types of graphs and their properties are described in the following Section.

Table 7.1: Network stats

Network name	Nodes	Edges	avg. k(v)	C'	global CC	CPL	Diameter
Amazon	334,853	889,705	2.66	0.3967	0.2052	11.96	46
Cit-HepTh	27,400	352,008	12.85	0.3139	0.1196	4.28	15
Email-EuAll	224,828	339,368	1.51	0.0791	0.0041	4.12	14
Enron	33,696	180,808	5.37	0.5092	0.0851	4.03	13
Flickr	105,722	2,315,901	21,91	0.0884	0.4015	4.33	9
Gnutella	62,561	147,875	2.36	0.0055	0.0039	5.93	11
Slashdot	82,165	504,185	6.14	0.0603	0.0241	4.07	13
WikiTalk	2,388,953	4,627,678	1.94	0.0526	0.0022	3.90	9
Youtube	1,134,530	2,984,125	2.63	0.0808	0.0062	5.28	21
Random3	50,000	150,000	3	0.0001	0.0001	6.25	11
Random5	50,000	250,000	5	0.0002	0.0002	4.95	8
Random10	50,000	500,000	10	0.0004	0.0004	3.89	6
Random25	50,000	1,250,000	25	0.0010	0.0010	3.04	4
Barabasi3	50,000	150,000	3	0.0006	0.0004	5.10	8
Barabasi5	50,000	250,000	5	0.0010	0.0007	4.28	6
Barabasi10	50,000	500,000	10	0.0017	0.0015	3.60	5
Barabasi25	50,000	1,250,000	25	0.0041	0.0038	2.92	4

7.1.1 Generated Networks

For our analysis we also used generated networks. We generated two kind of networks: Random based networks and scale-free networks (Barabasi Albert). For each type, we created four graphs, each with different average degree (3, 5, 10, 25). Every graph has 50,000 nodes.

Random

The Random network is simply generated in creating nodes and then picking random node IDs and connecting them with other random node IDs untill the specified amount of edges is reached.

Barabasi-Albert

The Barabasi-Albert is a model introduced by Barabasi and Albert [5] [4] to create scale-free networks. It imitates the behavior of growing social networks. To create a graph with the Barabasi-Albert model we start with a small amount of randomly connected nodes. Then we iteratively add nodes to the network. Then we connect the nodes with a preference for nodes with high degree. We choose to add an edge based on the Formula 7.1. If a random number r is smaller then the fraction of the degree of the destination node divided through the degree sum of all nodes in the graph, the edge will be added.

$$r \quad < \quad \frac{k(v_i)}{\sum_{j=0}^{n} k(v_j)} \tag{7.1}$$

7.1.2 SNAP

The Stanford Network Analysis Project (SNAP) provides a collection of large graphs, the Stanford Large Network Dataset Collection [1]. It includes social networks, web graphs, road networks, internet networks, citation networks, collaboration networks, and communication networks. All non generated networks that we used for our analysis are provided there.

Amazon

The Networks consists of pages crawled on the Amazon website. The collected websites are on basis of the Customers who bought this item also bought feature. If a product i is commonly co-purchased with another product j an undirected edge from i to j is part of the resulting graph. Each product category provided by Amazon defines a ground-truth community [1].

Cit-HepTh

High-energy physics theory citation network

The Arxiv HEP-TH (high energy physics theory) citation graph results from all the citations within a dataset of 27,770 papers with 352,807 edges of the e-print arXiv. If a paper i cites another paper j, the resulting graph contains an edge from i to j. It is a directed edge between i and j. All other citations from outside the dataset are not included into the resulting graph. The graph covers all papers in the period from Januar 1993 to April 2003 (about 124 months). Beginning with a few months of the inception of the arXiv, it represents the complete history of its HEP-TH section. In year 2003 the dataset was released as a part of the KDD Cup [1].

We converted this network into an undirected version.

EmailEuAll

The network is the result of all email data from a large European research institution. Between October 2003 to May 2005 (about 18 months) all information have been anonymized by the research institution. Each sent or received email was saved with its timestamp, its sender and the recipient of the email. Overall 3,038,531 emails between about 287,755 different email addresses. Note that we have a complete email graph for only 1,258 email addresses that come from the research institution. Although, there are 34,203 email addresses that both sent and received emails in the considered dataset. All other email addresses are either non-existing, mistyped or spam. In the resulting graph, each node corresponds to an email address. A directed edge between nodes i and j represents the fact that i sent at least one email to j [1].

We converted this network into an undirected version.

Enron

The Enron email communication network (originally released by William Cohen at CMU) covers all communication within a dataset of around half million emails. The Federal Energy Regulatory Commission made this dataset public and posted it to the web during its investigation. Each node of the network represents an email address and if an email address i sent at least one email to address j, the graph contains an undirected edge from i to j. Note that email addresses act as sinks and sources in the network as if they are non-Enron email addresses. Only communication with Enron email addresses are observed [1].

Flickr

By forming links between images sharing common metadata from Flickr, edges represent images with same location, submitted to the same gallery, group, or set, images sharing common tags, images taken by friends, etc. The images originally were collected from PASCAL, ImageCLEF, MIR, and NUS-wide [1].

Gnutella

The Gnutella graph results from a sequence of snapshots of the Gnutella peer-to-peer file sharing network from August 2002. Nine snapshots have been taken. Nodes represent hosts in the network and edges represent connections between the hosts of the Gnutella hosts [1].

Slashdot

Slashdot is a technology-related news website know for its specific user community. User-submitted and editor-evaluated current primarily technology oriented news are the main content of the website. In year 2002, Slashdot introduced a feature which allows users to tag each others as friends or foes. It is called Slashdot Zoo. The network contains links for friend/foe relationships between the users of Slashdot. It was obtained in February 2009 [1].

Wiki Talk

The WikiTalk network is based on Wikipedia, a free encyclopedia written collaboratively by volunteers arount the world. For each registered user, a talk page exists, that is editable by the user itself or other users. It can be used in order to communicate and discuss updates to various articles on Wikipedia. The network is the result of the latest complete dump of Wikipedias user talk pages from January 3rd, 2008. It contains all the users and discussions from the inception of Wikipedia till January 2008. Nodes represent Wikipedia users and a directed edge from a node i to another node j should be considered as the fact, that user i at least once edited the talk page of user j [1].

We converted this network into an undirected version.

Youtube

Youtube is a video-sharing web service that includes a social network with users forming friendships and groups. Such user-defined groups are considered a ground-truth communities. The whole dataset is provided by Alan Mislove et al. Each connected component in a group is a separate ground-truth community. Ground-truth communities with less than 3 nodes have been removed. The top 5,000 communities with highest quality are also provided. For the network, we provide the largest connected components [1].

7.2 Setup

For the analysis we used our implementation shown in Chapter 6. We analyzed 10 different non-revisiting algorithms (BFS, DFS, DFS_random, RWnr, MOD, GO, FFnr, FS, SS, US) on every network listed in Table 7.1. The series were generated with 20 runs per network instance, which means we sampled every network at least 20 times with every sampling algorithm. Every run constists of 200 batches with a batch size of $|B(t)| = \frac{n}{200}$. In every step we analyzed 6 different metrics: Extent, Degree Distribution, Clustering Coefficient, Sampling Modularity, Motifs and SSSP. We did not evaluate the Betweenness Centrality because the runtime of this metric was too high as it was the same with the All Pairs Shortest Paths. The All Pairs Shortest Path is approximated with the Single Source Shortest Path Recomputation metric, which we gave the parameter $l = \frac{|V|}{100}$. The Extent metric was calculated as Recomputation, the other metrics were calculated update based. All metrics except the SSSP are evaluated exactly and are no approximations.

We further created a set of series for analyzing the Start Node Selection Strategies. For those series we only used DFS, BFS, RWnr, GO and MOD. We analyzed all Start Node Selection Strategies. HRDS was analyzed with 5, 10, and 100 nodes, HRDSS with 5-5 and 100-25 nodes. For IS we just picked the node with the ID 10. The NVLS was started with the node value list of the local Clustering Coefficient and NVLS picked the node with the maximum. These series were only generated on the Enron graph.

Additionally we generated series for the analysis of the parameters for Frontier Sampling, Snowball Sampling and Forest Fire. These series were generated on three very differenz networks: Enron, Flickr and Cit-HepTh. We analyzed the sampling algorithms with 10 different

parameters on every graph. Frontier Sampling was started with 1-10 walkers, Snowball Sampling with l=1-10, and Forest Fire with a probability of 0.1-1.0.

Overall we generated and analyzed 43 series.

The runtimes of our analysis on the WikiTalk network was too hich so we skipped this network. On the same hand the runtimes of the motifs and SSSP metric on the Youtube and Flickr network was too high, so we did not analyze these two metrics on those networks.

7.3 Results

In this Section we are showing the results of our analysis. We evaluated the plots of all sampling algorithms regarding to the metrics Extent, DD, CC, SM, Motif and SSSP. The presentation of all plots would take too much space, therefore we are showing only the representative plots on a single network instance. The remaining plots are added, if the progress is remarkable. Our evaluation produced several findings which we explain step by step.

Finding 1: Our ordering of nodes does not play a role for the DFS performance

On our search for the reason of the bad performance of DFS we checked if our ordering of the nodes plays a role. Especially we were interested in the order in which DFS accesses the neighbors of a node and adds them to its queue. For this reason we implemented the DFS_random sampling algorithm, which randomizes the ordering of the nodes before they are added to the queue. Then we compared the two DFS versions.

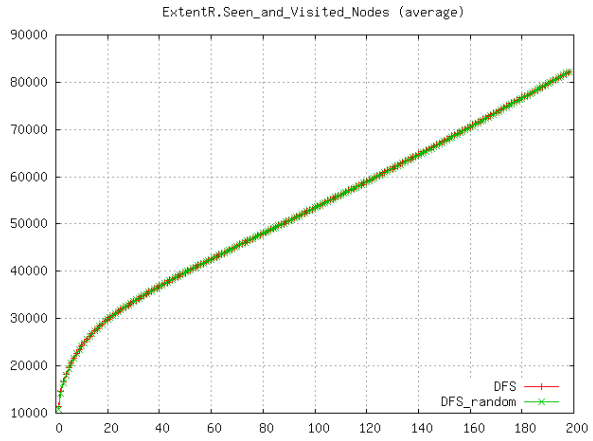

Figure 7.1: DFS / DFS_random comparison on Slashdot Extent

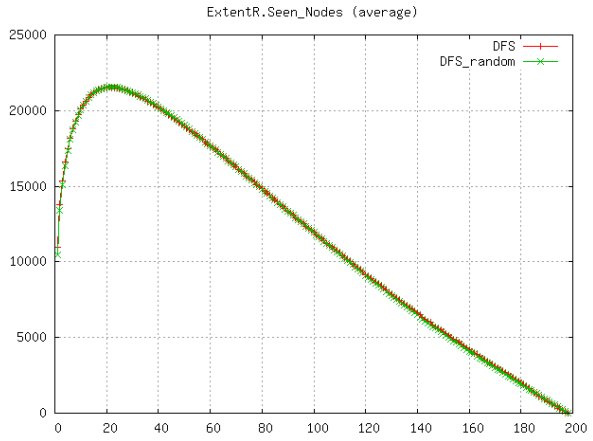

Figure 7.2: DFS / DFS_random comparison on Slashdot - Extent

As Figures 7.1,7.2,7.3, and 7.4 show, DFS and DFS_random behave exactly identical. Therefore the ordering of our DFS implementation did not play any role and we had to dig deeper for the reason of the bad performance.

Finding 2: The key of the DFS performance

One of the main reasons behind our analysis was to find out why DFS has such a poor performance regarding the MCNC problem and what is the difference between RWnr and DFS. Our

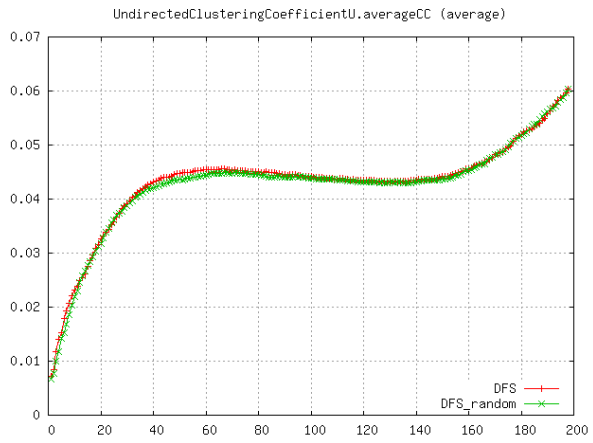

Figure 7.3: DFS / DFS_random comparison on Slashdot - Clustering Coefficient

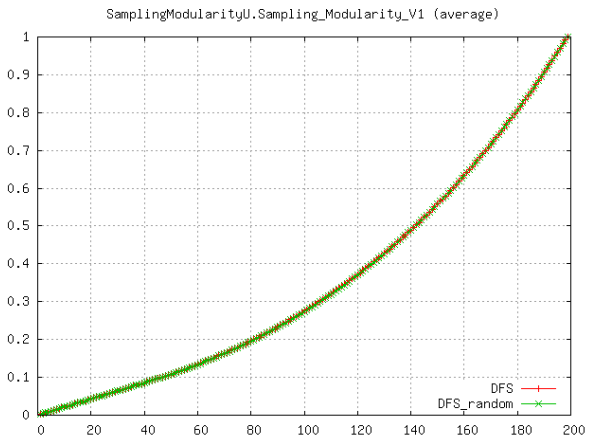

Figure 7.4: DFS / DFS_random comparison on Gnutella - Sampling Modularity

analysis showed that on many social networks, DFS performs worse than all other sampling algorithms regarding the MCNC problem, except from Uniform Sampling which is at the same level (See Figure 7.5). The Figure 7.6 show the significant difference between DFS, BFS and RWnr. Furthermore DFS strongly biases metrics like Motifs (See Figure 7.7), CC (See Figure 7.8) and especially CPL and Diameter (See Figures 7.9 and 7.10).

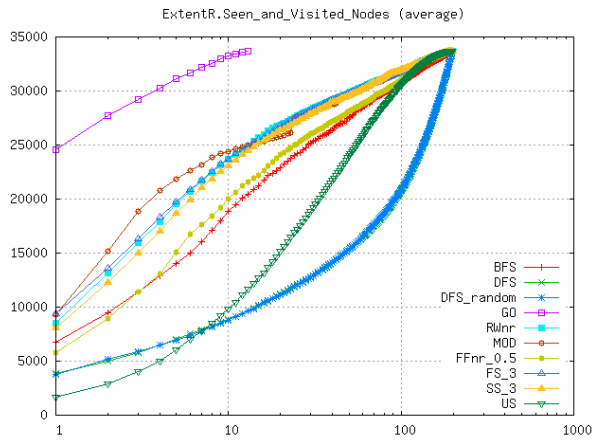

Figure 7.5: All SA on Enron - Extent

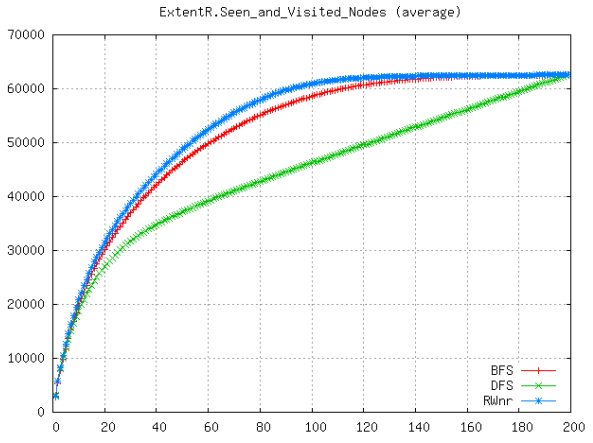

Figure 7.6: BFS / DFS / RWnr on Gnutella - Extent

The reason for this behavior is that the DFS sampling algorithm samples long chains of nodes with no further connections. This results in a very high CPL and Diameter. Due to that fact the sample does contain as few edges as possible and this results in a low Sampling Modularity V1 value (See Figure 7.11).

But where is actually the difference between RWnr and DFS? Tim Grube [9] stated that the only difference between these sampling algorithms is their behavior if they reach an dead end, a node which only has neighbors that are already visited. Our analysis showed that this is clearly wrong because both, DFS and RWnr sampled graphs like the Enron graph without running into

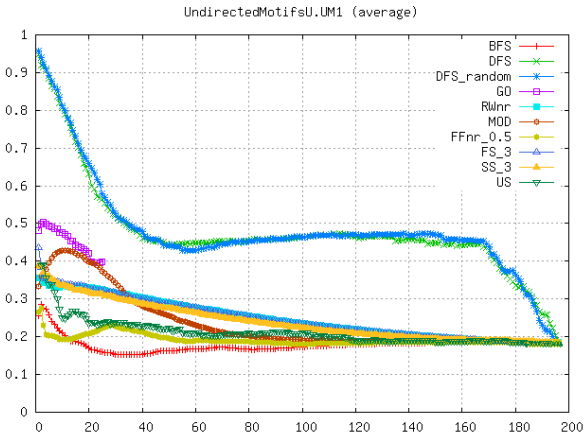

Figure 7.7: All SA on Cit-HepTh - Motif 1

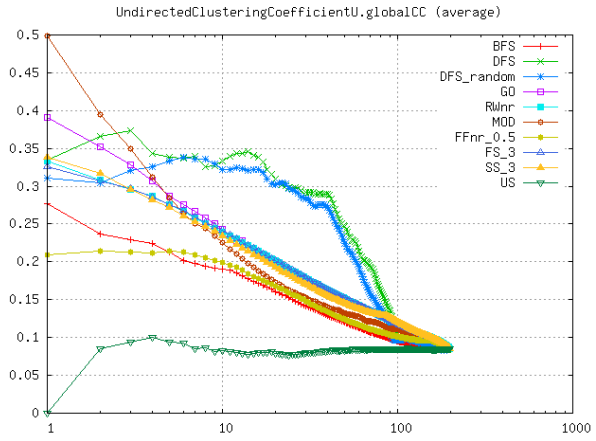

Figure 7.8: All SA on Enron - Clustering Coefficient

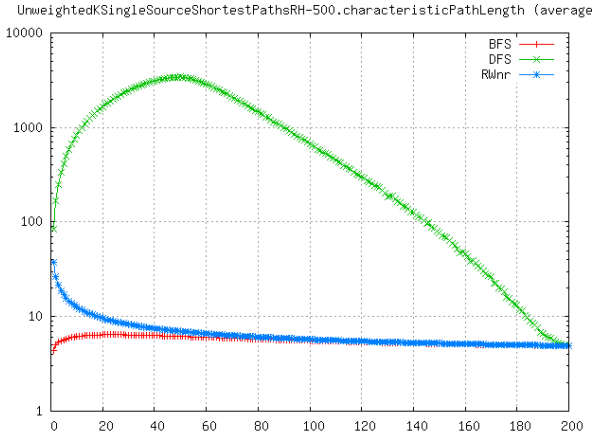

Figure 7.9: BFS / DFS / RWnr on Random5 - CPL

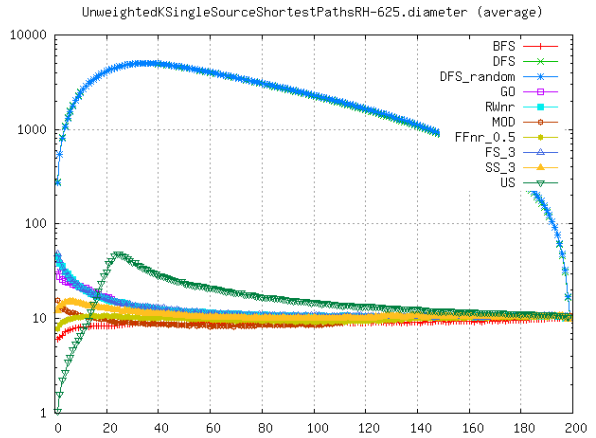

Figure 7.10: All SA on Gnutella - Diameter

an dead end once. The difference of the behavior lies in the use of a queue. We explain this with an example. Imagine the situation shown in Figure 7.12.

Both, RWnr and DFS, have visited the nodes 1 and 2. The queue of the DFS algorithm looks like {3, 5 ,4}. Lets assume that RWnr randomly picks node 3 in the next step and visits it. DFS will do the same, because node 3 is the next node in its queue. The DFS algorithm will now add all unvisited neighbors to its LIFO queue. The only unvisited neighbors, node 4, is already in the queue (because it is also a neighbor of the previously visited node 1) and will not be added. In the next step RWnr has to visit node 4 because it is the only unvisited neighbor of node 3. DFS instead will visit the node 5, because it is the next node in the queue.

This is the core of the problem. DFS will always avoid visiting nodes that have connections to already sampled nodes (like node 4) and therefore DFS will avoid clusters and well connected communities. This leads to a very high CPL and Diameter, as well as a bad performance in the Extent metric.

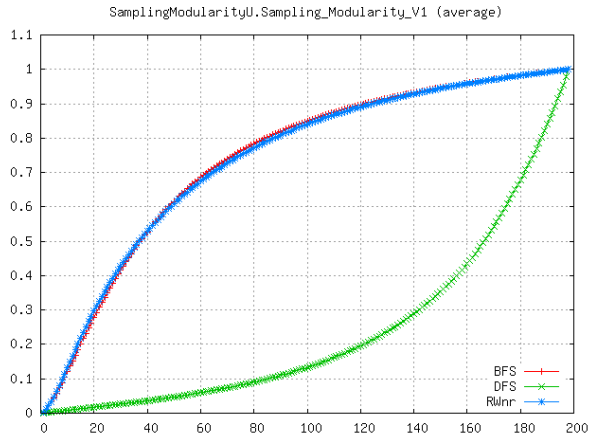

Figure 7.11: DFS / BFS / RWnr on Slashdot - Sampling Modularity

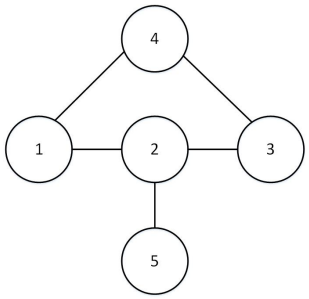

Figure 7.12: Graph example for DFS and RWnr

Finding 3: The performance of MOD is strongly related to the structure of the network

Our analysis showed that the performance of MOD can be very variable and it depends on the structure of the network. On the Slashdot graph for example MOD outperforms every other sampling algorithm except GO, which was a one hop look ahead (See Figure 7.13).

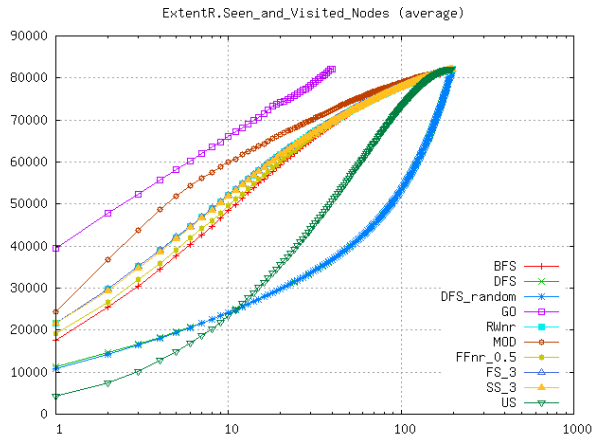

Figure 7.13: All SA on Slashdot - Extent

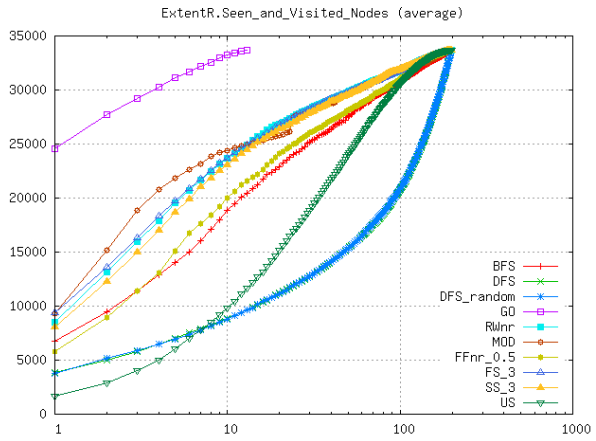

Figure 7.14: All SA on Enron - Extent

On the Enron graph the performance is good in the beginning but then it drops below the performance of other sampling algorithms (See Figure 7.14). The performance of MOD on the Amazon graph is worse than most of the other sampling algorithms (See Figure 7.15).

The Figure 7.16 shows us the core of the problem. GO visits and sees far more nodes while it samples far less edges than MOD. MOD instead samples an high amount of edges. This happens because MOD always visits the node with the highest number of edges to nodes in the sample and therefore maximizes the amount of edges sampled. This behavior leads MOD to prefer clustered and well connected communities but the algorithm tends to circle in these communities. The approximation of the maximum uncovered degree with the maximum observed degree is not optimal. The focus on well connected neighborhoods can be a good approach on social

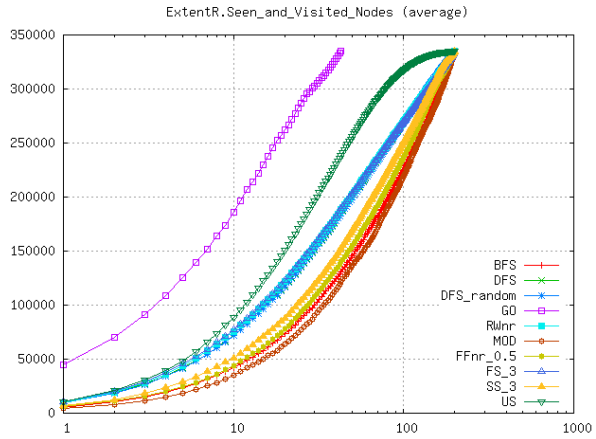

Figure 7.15: All SA on Amazon - Extent

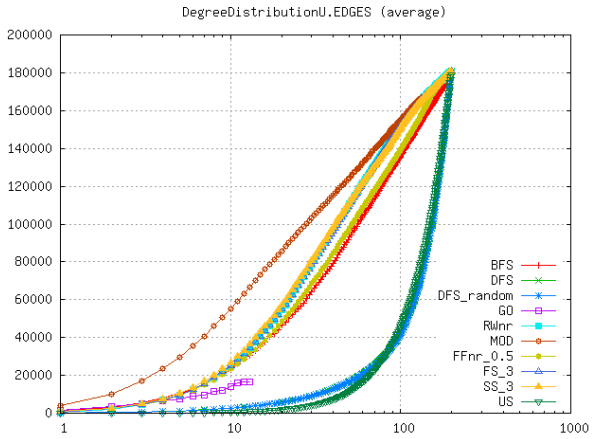

Figure 7.16: All SA on Enron - Edges

networks but it has its disadvantages. In any way the performance is strongly related to the structure of the network.

Finding 4: Uniform Sampling perfectly approximates the Motif metric

Our intention of analyzing Uniform Sampling was to have a good baseline for the extent metric. But our analysis shows that US converges very slow and too variable regarding the MCNC problem to work as a baseline. But the analysis also shows that the Motif metric on samples generated with the Uniform Sampling algorithm converge to the original value very fast. Most of the series show that US reaches the original value before the 10th batch, which means the algorithm sampled less than 10% of the whole network. But even before the 10th batch, the value is far more close to the original value than the values of any other sampling algorithm (See Figures .. and ..).

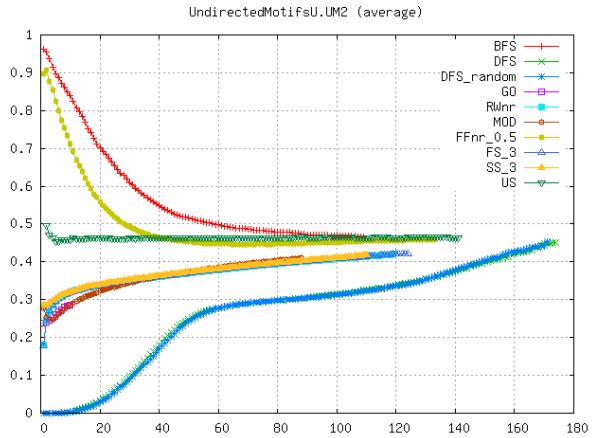

Figure 7.17: All SA on Barabasi10 - Motif 2

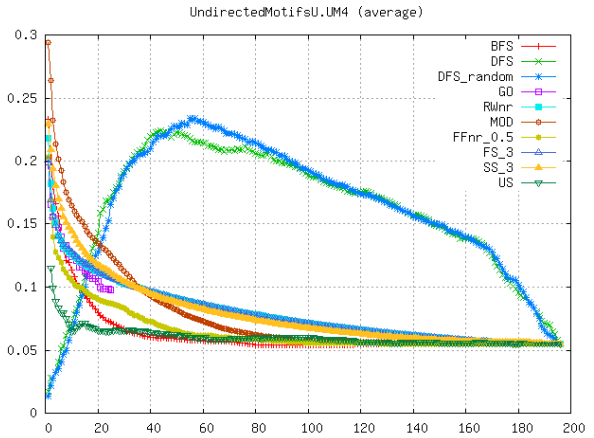

Figure 7.18: All SA on Cit-HepTh - Motif 4

The reason for this is the that Uniform Sampling picks nodes completely random and has a perfect fair selection. Every node is picked with the same probability. Therefore we get a very broad and differentiated picture of the graph.

Finding 5: Influence of the Start Node

Another important aspect for us was to find out if the selection of the start node has influence on the behavior of the sampling algorithm. Our analysis shows that the start node can have an influence on the performance of the sampling algorithm but it strongly depends on the sampling algorithm. GO, MOD and RWnr do not show any difference in their performance or behavior (See Figures 7.19 7.20). The DFS algorithm shows a small impact with different Start Node Selection Strategies but it is not significant (See Figure 7.21).

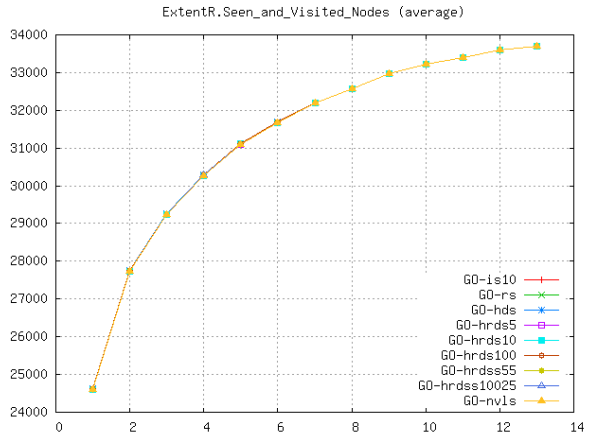

Figure 7.19: GO with SNS on Enron - Extent

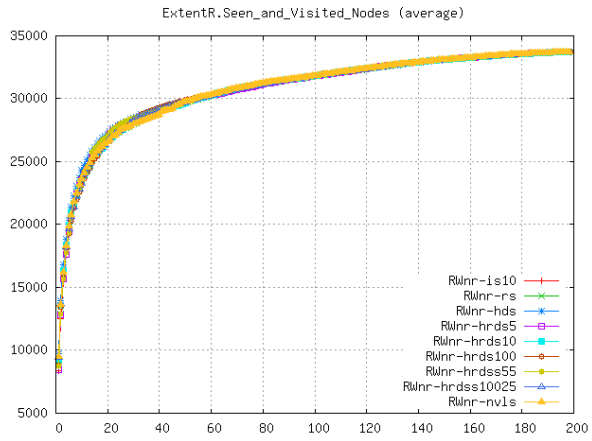

Figure 7.20: RWnr with SNS on Enron - Extent

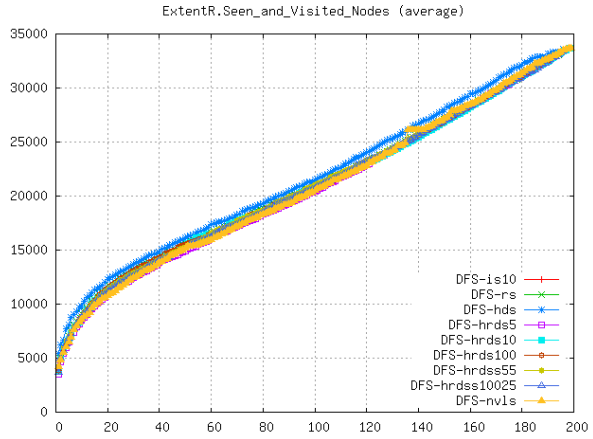

Figure 7.21: DFS with SNS on Enron - Extent

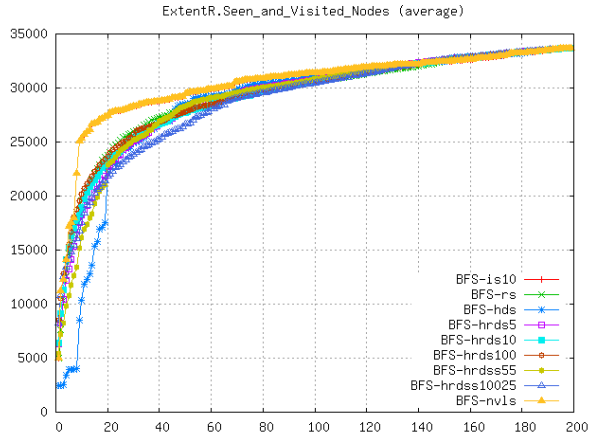

Figure 7.22: BFS with SNS on Enron - Extent

BFS instead shows an remarkable impact. In Figure 7.22 you can see the influence of the different strategies. Especially the Node Value List Selection has a significant effect on the performance of BFS in the first half of the run.

The reason for the impact on BFS is that BFS is the only sampling algorithm which directly visits all neighbors of the start node. Therefore BFS has the biggest profit of many influential neighbors. This explains the impact of the NVLS, which selects the node with the highest average Clustering Coefficient as start node. This node is well connected and center of a well connected

community or at least neighborhood. BFS directly profits from this neighborhood as it can see many nodes with few steps.

Finding 6: The amount of walkers does not influence performance of Frontier Sampling

Our analysis shows that the parameter of FS have no significant influence on the performance of the sampling algorithm on the Extent metric and towards the MCNC problem. The Figures 7.23 and 7.24 show that there is hardly any difference between the algorithms.

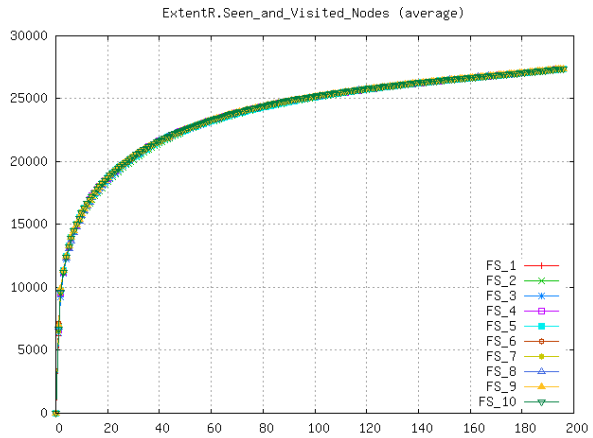

Figure 7.23: FS 1-10 on Cit-HepTh - Extent

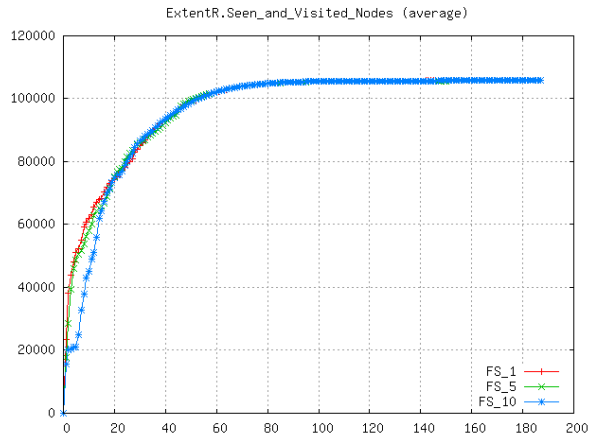

Figure 7.24: FS 1-10 on Flickr - Extent

The reason for this is that Frontier Sampling picks the next node random based and therefore it behaves like a Random Walk. The amount of walkers do not change the characteristic of the node selection, they just spread this behavior on more positions in the network.

Finding 7: ForestFire performs slightly better with lower p values

The analysis of the Forest Fire algorithm with different parameters p shows that a lower p value is an advantage regarding the MCNC problem. The Figures 7.25 and 7.26 show a small but measureable tendency towards a better performance with lower probability p.

The cause of this is that we do not have to visit all neighbors to see all or most of them. In most of the cases it is enough to visit the center of a cluster or a few well connected nodes.

Finding 8: Snowball Sampling performs better with lower l values

The analysis of the Snoball Sampling algorithm with different parameters l shows that a lower l value is an advantage regarding the MCNC problem. It is exactly the same cause as for the Forest Fire. The Figure 7.27 shows that the trend is not significant but measureable.

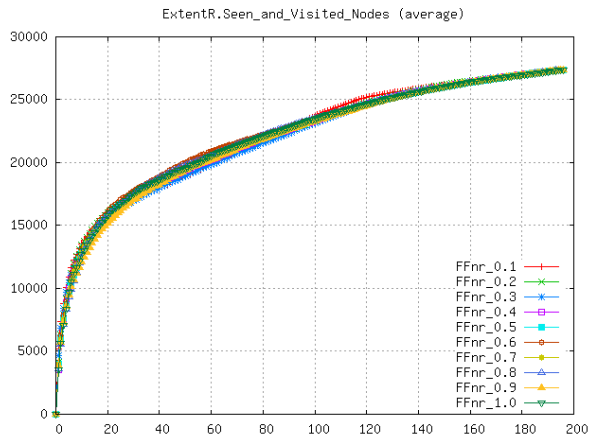

Figure 7.25: FF 0.1-1.0 on Cit-HepTh - Extent

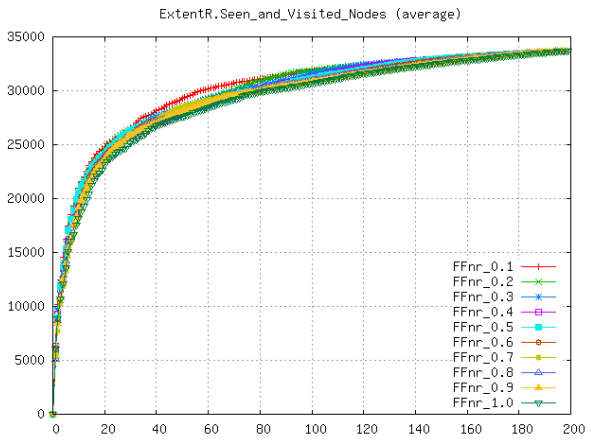

Figure 7.26: FF 0.1-1.0 on Enron - Extent

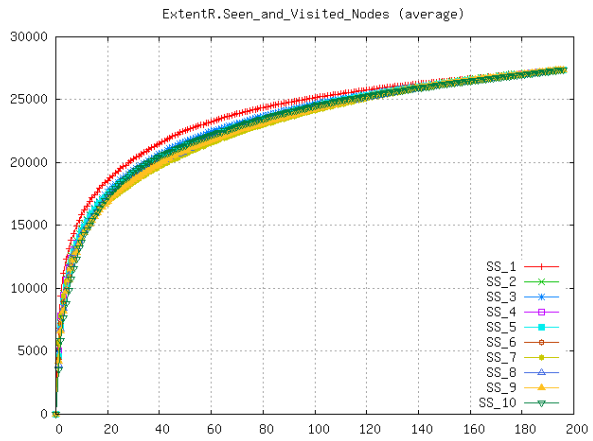

Figure 7.27: SS 1-10 on Cit-HepTh - Extent

8 Summary, Conclusion and Outlook

8.1 Summary

In the previous Chapters we dealt with the problem of understanding how and why sampling algorithms have different performances in analyzing networks. Our focus lies on the performance towards the Maximum Connected Network Coverage (MCNC) problem. The MCNC problem deals with the challange to explore and see the whole network with as least steps as possible. A node is marked as seen if a neighbor of the node is visited. The MCNC problem is of special interest regarding the growing importance of (online) social networks. Often we do not have detailed informations about those networks and therefore the analysis of sampling algorithms that do not have previous knowledge of the network were on focus of this work. The MCNC and related problems as well as possible solutions for it are the basis of several works [10] [16] [3].

For our work we defined a set of network measures, which we thought might give us an hint on which the performance of different sampling algorithms depend. These metrics are measured on the samples generated by the sampling algorithms. The set of sampling algorithms that we consider in this thesis are mainly inspired by Avrachenkov et Al. [3] and Tim Grube [9]. We categorized the algorithms into five groups: Random Walks, DFS-based algorithms, BFS-based algorithms, maximum coverage algorithms and Uniform Sampling. The group of Random Walks consits of the classical Random Walk with and without revisiting as well as the Frontier Sampling. DFS-based algorithms are only the Depth-First-Sampling algorithm. The DFS plays a special role in this work because one of our main questions was why the DFS has such a poor performance on social networks regarding the MCNC problem. The group of BFS-based algorithms consists of the BFS and BFS related algorithms like RDS, SS, and FF with and without revisiting. The maximum coverage algorithms are a group of sampling algorithms specially designed for the MCNC problem. These algorithms are MOD, GO, and MEUD. The last group only represents one sampling algorithm, the Uniform Sampling. This algorithm was intended to be the baseline for some metrics. Another question which we want to answer with this work was the influence of the start node for sampling processes. Therefore we designed a set of different Start Node Selection Strategies.

For our analysis we used the DNA framework, a java framework that was developed at the TU Darmstadt. The framework was designed to dynamically analyze networks. We extended the framework for the use with sampling algorithms. In our analysis we measured the metrics on the samples generated with the different sampling algorithms. We analyzed the behavior and performance of the algorithms ony many different graphs (technical, social, and generated networks). Our selection of graphs was inspired by Avrachenkov et Al. [3]. In the next Section we summarize the findings of our analysis.

8.2 Conclusion

Our analysis had serveral results. First of all we found the reason why DFS performs bad on social networks. DFS strictly avoids nodes that already have a connection to the sample and as a result DFS avoids clusters. The cause of this is the LIFO queue. This results in the bad performance in exploring social networks, since these networks are often highly clustered. In the same way DFS avoids clusters, MOD prefers them. This behavior leads to good performance on social networks but on a poor performance on technical or non-social networks. Further our analysis showed that the start node does not have any or very small influence on the performance of most sampling algorithms. BFS is the only sampling algorithm whose performance got influences by the start node, because BFS directly visits all neighbors of this node and therefore this node has an impact on the performance.

Our analysis of the parameters of Forest Fire and Snowball Sampling showed that it is an advantage regarding the MCNC problem to more behave like an RW and less like a BFS. This is due to the fact that we do not have to visit all neighbors to see all or most of them.

8.3 Outlook

There are still several open questions related to this topic. First of all we did not analyze the impact of directed graphs on our results. This would be an open topic for further works.

Furthermore we did not analyze the influence of Start Node Strategies on other non-social networks, which could be of interest too.

As we analyzed the DFS and DFS_random, we came to the question what influence a willfull order of the nodes added to the queue might have. We could imagine a DFS which orders the nodes after their degree before adding them to the queue.

As mentioned before we did not had the chance to analyze the Betweenness Centrality metric, since it takes too long to compute it. We could imagine a approximation for this metric which only uses the top l nodes for the computation (like the SSSP approximation for APSP).

List of Figures

List of Tables

Bibliography

[1] Stanford large network dataset collection. Website. `http://snap.stanford.edu/data/index.html`; Accessed: 2014-06-10.

[2] Jac M Anthonisse. The rush in a directed graph. *Stichting Mathematisch Centrum. Mathematische Besliskunde*, BN 9/71:1–10, 1971.

[3] Konstantin Avrachenkov, Prithwish Basu, Giovanni Neglia, Bruno Ribeiro, and Don Towsley. Online myopic network covering. *arXiv preprint arXiv:1212.5035*, 2012.

[4] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[5] Eric Bonabeau and Albert-Laszlo Barabasi. Scale-free networks. *Scientific American*, 289(5):60, 2003.

[6] Deepayan Chakrabarti and Christos Faloutsos. Graph mining: Laws, generators, and algorithms. *ACM Computing Surveys (CSUR)*, 38(1):2, 2006.

[7] Linton C Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977.

[8] Leo A Goodman. Snowball sampling. *The annals of mathematical statistics*, pages 148–170, 1961.

[9] Tim Grube. Sampling-based network analysis. Master's thesis, Technische Universität Darmstadt, 2013.

[10] Sudipto Guha and Samir Khuller. Approximation algorithms for connected dominating sets. *Algorithmica*, 20(4):374–387, 1998.

[11] Hyoungshick Kim, Konstantin Beznosov, and Eiko Yoneki. Finding influential neighbors to maximize information diffusion in twitter. In *Proceedings of the companion publication of the 23rd international conference on World wide web companion*, pages 701–706. International World Wide Web Conferences Steering Committee, 2014.

[12] Hyoungshick Kim and Eiko Yoneki. Influential neighbours selection for information diffusion in online social networks. In *Computer Communications and Networks (ICCCN), 2012 21st International Conference on*, pages 1–7. IEEE, 2012.

[13] Vaishnavi Krishnamurthy, Junhong Sun, Michalis Faloutsos, and Sudhir Leslie Tauro. Sampling internet topologies: How small can we go? In *International Conference on Internet Computing*, pages 577–580. Citeseer, 2003.

[14] Maciej Kurant, Athina Markopoulou, and Patrick Thiran. On the bias of bfs (breadth first search). In *Teletraffic Congress (ITC), 2010 22nd International*, pages 1–8. IEEE, 2010.

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

[16] Arun S Maiya and Tanya Y Berger-Wolf. Benefits of bias: towards better characterization of network sampling. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 105–113. ACM, 2011.

[17] Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

[18] Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

[19] Bruno Ribeiro and Don Towsley. Estimating and sampling graphs with multidimensional random walks. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 390–403. ACM, 2010.

[20] Bruno Ribeiro, Pinghui Wang, Fabricio Murai, and Don Towsley. Sampling directed graphs with random walks. In *INFOCOM, 2012 Proceedings IEEE*, pages 1692–1700. IEEE, 2012.

[21] Benjamin Schiller, Dirk Bradler, Immanuel Schweizer, Max Mühlhäuser, and Thorsten Strufe. Gtna: a framework for the graph-theoretic network analysis. In *Proceedings of the 2010 Spring Simulation Multiconference*, page 111. Society for Computer Simulation International, 2010.

[22] Benjamin Schiller and Thorsten Strufe. Dynamic network analyzer - building a framework for the graph-theoretic analysis of dynamic networks. In *Summersim*. SCS, July 2013.

[23] Daniel Stutzbach, Reza Rejaie, Nick Duffield, Subhabrata Sen, and Walter Willinger. Sampling techniques for large, dynamic graphs. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–6. IEEE, 2006.

[24] Daniel Stutzbach, Reza Rejaie, Nick Duffield, Subhabrata Sen, and Walter Willinger. On unbiased sampling for unstructured peer-to-peer networks. *IEEE/ACM Transactions on Networking (TON)*, 17(2):377–390, 2009.

[25] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440–442, 1998.