Social Network Analysis: Betweenness Centrality Estimation via Sampling and Correlation with Dijkstra and Brandes Algorithm

Social Network Analysis Course Project Paper

Athanasios Agrafiotis
LIACS, Leiden University
a.agrafiotis@umail.leidenuniv.nl

Yuchi Zhang
LIACS, Leiden University
y.zhang.48@umail.leidenuniv.nl

ABSTRACT
This paper has as a main task to compare two different algorithms that computes the betweenness centrality for the concept of computation time and to calculate the betweenness centrality using two different techniques one of Freeman and Brandes. Calculate the betweenness centrality in a real-life social networks represents a significant challenge due to the size of network's data. Previous approaches have as a contribution to develop efficient algorithms and to combine them , Dijksta and Breadth First Search. In this paper, we consider these factors significant and motivate the use of specific algorithms through a careful investigation of results. As it was mentioned earlier in the first sentence the computation of betweenness centrality between the Dijkstra with Freeman and Brandes differs. The Freeman formula consider the fraction of times that a node appears in all the shortest paths divided by the summation of shortest paths equation 3. Brandes consider the geodesic path of a distance which is multiplied by the all the shortest paths divided by the summation of shortest paths equation 1.
KEYWORDS
Betweenness centrality, microscale analysis, Algorithms

1. INTRODUCTION

In network analysis one of the problem is to examine and to determine the properties of a specific node in the network. Microscale analysis in networks are the closeness, betweenness and degree centrality [4]. Betweenness centrality has been one of the most important properties in recent researches such as the community detection [5]. In social networks, algorithms can efficiently and effectively find nodes which play crucial roles in a network. In addition betweenness centrality can be an effective way to optimize network selection strategies to identify communities [5].
Most of scientists demands algorithms that can traverse a graph efficiently in time. Graph traverse techniques, receive the most attention in calculation of microscale analysis. Microscale analysis represents an analysis that identify highly important nodes in a network. There are a huge number of algorithms proposed. Dijkstra's algorithm is a known single source shortest path algorithm for graphs and follows a greedy strategy. Greedy strategy as the algorithm selects the optimal move to move though the network. Single source path computes the shortest path from a given node to a destination node. Next the summation of shortest paths are used to formulate the Betweenness score equation. We recognize that this is not the first paper using graph algorithm in deterministic environment prior knowledge of the network. Applies a greedy and traversing techniques to calculate betweenness centrality scores, but we demonstrate in our research one of the most important aspects by comparing a Dijkstra algorithm and a combination of BFS and Dijkstra algorithm.
High betweenness centrality nodes/vertices can identify important edges between users and can spread the disease. An example of a real world problem that betweenness centrality can solve is to identify high important nodes in a graph that can spread disease through the network. A real-world problem it can be the flu and cold. Another example is to identify roads with high betweenness centrality. Interesting places for people to make connection or high interesting place for infrastructure (meeting place).
In this paper it is recommended an approach to compare algorithms. It is presented two existing algorithms implementation of Dijkstra and Brandes algorithm. The biggest advantage of our contribution is to compare two existing techniques that computes the betweenness centrality of the graph in terms of computation time.
A portion of the graph which is at least eighty percent less than the whole graph would be a good solution with lower error. To summarize, our contribution are mainly twofold. First the implementation of two different algorithms next one greedy and traverse technique combined recommended by et al Brandes [2]. Second and most important we recommend a shortest path computation based on a sampling method . Our observations on the network depicts that the problem of betweenness centrality for a large networks can be solved in $O(nm)$ time for unweighted graphs and $O(nm + n^2 log(n))$ for weighted graphs
The rest of the paper is organized as follows. In Section 2 we formulate the problem of betweenness Centrality. In Section 3 we introduce some previous work and how it is related to our work. In Section 4 how algorithms work and how we will use them for our experiment and which are the factors that

SNACS '20 Social Network Analysis for Computer Scientists, Master CS, Leiden University (`liacs.leidenuniv.nl/~takesfw/SNACS`).

we need to consider. In Section is our datasets, In Section 6 the evaluation of our experiment and finally in Section 7 conclusion and future work. Cited papers are referenced in the Section 8.

2. PROBLEM STATEMENT

This paper measures the correlation of betweenness centrality score using two approaches a traverse technique and a greedy technique. The problem question of this paper, how does the betweenness centrality score of Freeman and Brandes are correlated. How much portion of the graph is needed to calculate the betweenness centrality.
Besides, we learn a random walk algorithms from a paper [8]. We do not focus on estimating the betweenness centrality for all the graph. Instead, we keep sampling the graph and accurately estimate the top-k nodes with the highest betweenness centrality from a small sample of a graph. The proposed algorithm firstly obtains a small sample that includes many of top-k nodes with the highest betweenness centrality via a random walk on a social network. Then, we obtain unbiased estimates of the ego betweenness centrality (shown in section 4) of sampled nodes and approximate the top-k nodes with the highest betweenness centrality as the top-k nodes with the highest estimated ego betweenness centrality. Though ego Betweenness centrality differs from the Brandes and slightly from Freeman, but it can estimate top-k nodes efficiently equation 2. The proposed estimator efficiently estimates the ego betweenness centrality of each sample without additionally sampling the graph data by utilizing the neighbor data of the previous and the next samples [8]. The experiments using real social network datasets show that the proposed algorithm estimates more accurately the top-k nodes with the highest betweenness centrality than existing algorithms when the sample size is small. For a graph given a set of vertices which defined as $u \in V$ and edges as $m \in E$. The distance represents the summation of the edge weights from the start vertex to the destination vertex. The shortest path $\sigma_{u,v}$ represent the minimum distance from every distance $D_G u, v \in V(G)$ of u to v, in a given graph G. The predecessor maintains the shortest paths. The dependency notated as $\delta_{s*}(e)$ represents the number of links in the shortest path between e and s . The betweenness score of a weighted graph defined as the score of a given vertex u on shortest paths of v to s defined as follow:

$$s : P_v(u) = \{u \in V : \langle u, v\rangle \in E, d(v,u) = d(v,u) + w(u,v)\} \tag{1}$$

The ego betweenness centrality represents the fraction of the times that u appears in shortest path of uj and uk as $\sigma_{j,k(i)}$ divided by the shortest path of uj and uk as $\sigma_{j,k}$. Notation used from paper [8].

Ego betweenness centrality equation:

$$eBC(i) = \sum_{v,u \in V} \frac{\sigma_{j,k(i)}}{\sigma_{j,k}} \tag{2}$$

Freeman betweenness centrality equation:

$$BC(e) = \sum_{v,u \in V} \frac{\sigma_{u,v(\epsilon)}}{\sigma_{u,v}} \tag{3}$$

Brandes betweenness centrality equation:

$$\delta_{s*}(e) = \sum_{e:u \in P_s(e)} \frac{\lambda_{su}}{\lambda_{se}}(1 + \delta_{s*}(e)) \tag{4}$$

$$BC(e) = \sum_{s \neq u \neq t \in V} \delta_{st}(e) \tag{5}$$

The time complexity of a Dijkstra algorithm is O(m*nlog(n)), the time complexity of the combination of Breadth First and Djikstra as proposed by Brandes paper [2] is O(nm).

3. RELATED WORK

In 1971 and 1977 Anthonisse and Freeman introduced the centrality measures, determining betweennness centrality score. Geodesic represents the shortest path of all distances given a pair of nodes on a graph. In related work of David Bader et. al. [2]. Breadth first search was the first approach for computing the betweenness centrality. The approach represents a combination with adaptive sampling technique for an undirected graph. Dijkstra algorithm in Wei Peng et al. [9]. Another approach is the Bellman-Ford algorithm is proposed for networks with negative weights [1]. For dynamic shortest path problems the recommended approach were by Rodity and Zuid [10].
All the related work that has been introduced for betweenness centrality is based on comparing different algorithms,which compute the betweenness score using different equations of Freeman 3 and Brandes 1. Until now there is no record of such a research.

4. ALGORITHMS

The algorithm of breadth first search represents a traverse technique that has been used to explore nodes and edges of a graph, The time complexity differs based on the implementation. It starts from a current node u and next explore all it's neighbor. After it's neighbor has been explored the node u pushed to the S list as visited. The distance between the node (u) and it's neighbor is measured. If the distance of neighbor w has a value lower than zero then the neighbor w is added to the Q. If the neighbors of node (u) has value lower than the current distance are dismissed otherwise considered as shortest path and added to the list of shortest paths $P[w]$. $P[w]$ counts the times that a node appears in a shortest path. At last the δ represents as one if a node appears at least once in a shortest path we have not met another contribution for the same subject until now.
Predecessor finds the Shortest paths for selected node to other vertices. In each step calculates the dependency of s. To reduce time it calculates the shortest path from s to t. And it finds the betweenness centrality of all nodes u that belongs to that shortest paths.

Dijkstra algorithm 2 calculates the shortest path between two vertexes as follow. The following sets are used: S rep-

Algorithm 1 BFS Brandes-Betweenness Centrality

$S \leftarrow 0$
$d[s] \leftarrow 0$
$\sigma[s] \leftarrow 1$
$P[w] \leftarrow []$
$Q :=$ `a queue data structure, initialized with s`
while $Q \neq \emptyset$ **do**:
`dequeue` $u \leftarrow Q$
`push` $u \rightarrow S$
  for neighbor w of u **do**:
    if $d[w] < 0$ **then**
      `enqueue` $w \rightarrow Q$
      $d[w] \leftarrow d[u] + 1$
    end if
    if d[w] = d[u] + 1 **then**
      $\sigma[w] \leftarrow \sigma[w] + \sigma[u]$
      `append` $u \rightarrow P[w]$
    end if
  end for
end while
$d[w] \leftarrow 0, v \in V$
while S `not empty` **do**
  $w \leftarrow S$
  for V `in` $P[w]$ `do` $\delta[v] \leftarrow \delta[v] + \frac{\sigma[v]}{\sigma[w]} \cdot (1 + \delta[w])$; **do**
    if $w \neq s$ `then` $C_B[w] \leftarrow C_B[w] + \delta[w]$ **then**
    end if
  end for
end while

resents the visited and Q the unvisited nodes, respectively. The algorithms starts by setting the *pred* and *dist* as empty. Similar with BFS deque a node u from Q queue. Next all the edges of this nodes are explored and then they get insert in the list S. The relax function represents the $FS(u)$ which test an edge (u, v) if the shortest path of v it can improved. It tests each neighbor of the node u by comparing the distance(weight) with all the other neighbors. If weights are smaller than the previous then we add the edge to the shortest path.

Algorithm 2 Dijkstra-Freeman Betweenness Centrality

$dist[s] := 0; pred[s] := NULL; S := \emptyset$
for $u \in V \setminus \{s\}$ **do**:
  $dist[u] := +\infty; pred[u] := NULL;$
  $Q \leftarrow V$
end for
while $Q \neq \emptyset$ **do**:
  $u := extract\_min(Q);$
  $S := S \cup \{u\}$
  for $u \in FS(u)$ **do**
    if $dist[u] > dist[u] + w_{uv}$ **then**
      $dist[u] := dist[u] + w_{uv}$
      $pred[u] := u;$
    end if
  end for
end while
  $BC(u) = \frac{\sigma(\texttt{total number of u node appears} \in S)}{\texttt{total number of shortest paths } S}$

The Random walk generates a sample. Then we use special algorithm to calculate betweenness centrality for these samples and estimate the top kth nodes. In the algorithm. We sample indices and their neighbors of nodes by performing a random walk on a graph G. In a random walk, a walker repeatedly moves to a randomly selected neighbor. The transition probability of node vi to node vj in a random walk is defined as. Where $|d_i|$ is the number of the neighbors:

$$\begin{cases} \frac{1}{d_i}(v_j \in N(i)) \\ otherwise \end{cases} \tag{6}$$

Let R = $x_s{}_{s=1}^{r}$ be a sequence of indices of r sampled nodes via random walk, where x_s denotes an index of the sth sampled node. Let Pr[A] denote the probability that event A occurred. After many steps of a random walk, the probability Pr[xr = i] converges to a certain value. This is core algorithm of this proposed algorithm
Then, we define the set of the ordinal numbers of a sample sequence between 2 and $r-1$ where a node v_i is sampled as:

$$I(i) = s\, x_s = i, 2 <= s <= r-1$$

. For each $s \in I(i)$, we define the proportion of the shortest paths that pass through $v_{x_{s-1}}$ and $v_{x_{s+1}}$ as a variable $\phi s(i)$ as follows:

$$\begin{cases} \frac{1}{N(x_{s-1} \cap N(x_{s+1})} \\ 0 \end{cases} \tag{7}$$

Finally, we define the estimate of the ego betweenness centrality eBC of a node v_i as the average of $\phi s(i)$ that is weighted with d_i^2 to remove the sampling bias due to a random walk:

$$eBC(i) \begin{cases} \frac{1}{I(i)} * \sum_{s \in I(i)} d_i^2 \phi_s(i) \\ 0 \end{cases} \tag{8}$$

It is remarkable that we can obtain an unbiased estimator of the ego betweenness centrality, that needs the neighbor data in the calculation, without additionally sampling the neighbor data. The proposed estimator avoids additionally sampling the neighbor data by calculating the proportion of the shortest paths that pass through each sample between the previous and next samples, $\phi s(i)$.

The running time of the proposed algorithm with r samples is $O(r_{dmax})$, because the running time of computing $\phi s(xs)$ for each sample vx.

5. DATASETS

Department of Energy US(United States) 2010 each node represents regions that are supplied by electricity "Highways".
Deezer Europe dataset contains data. Nodes are users that follow artists and edges relationships between them. (binary classification).
musae facebook is a dataset that contains official facebook pages and links are mutual likes between sites. Category of the facebook pages are politician, govermental organizaions, tv shows and companies (multiclass classification).
LastFM asia social network : The corresponding dataset contains users and edges that are relationship between users.
Arxiv GR-QC: represents a collaboration network from the

scientific cover. Nodes are authors and the co-authored are edges. The dataset contains small subgraphs.
url:https://snap.stanford.edu/data/ [6]
Buzznet[11]: A social media network designed for sharing

	Nodes	Edges
US power grid	3907	10000
Deezer Europe	7309	10000
Facebook musae	2909	10000
LastFm Asia	3473	10000
Collab. Arxiv	2524	10000

Those data are proposed by Stanford to be used for research on Social network analysis. The data are not biased based for the techniques that this paper presents [3]. The labels of the data are in binary form which are better for research.

6. EXPERIMENTS AND RESULTS

The suggested approaches will be tested and verified based on the exact centrality scores of nodes of the whole graph. The variation score of nodes are represented in a scatter plot visualizing the thirty nodes with the highest centrality score of the dataset of US power grid, Deezer Europe, facebook musae, LastFm Asia, Collaboration Arxiv.

We evaluate these algorithms using real social network datasets from the following two viewpoints:

(1) Estimation accuracy: We show that the proposed algorithm improves the estimation accuracy of the top-k nodes with the highest betweenness centrality with the small sample size.

In figure 1, 2, 4, 3. 5 are display the betweenness centrality score of top thirty nodes using the algorithm of Brandes. In all figures the distribution is heavy tailed.
In figure 6 is presented the betweenness centrality score using Djikstra algorithm and Freeman formula. In contrast with the index of nodes that the figure display. In figure 2 node with index *1* has lower betweenness centrality score than figure 6 which it's score is really high. The node with index *1901* is display in the first position with 2 in contrast with 6 which is in the position sixteen.
The figure 7 it is display the absolute error of Brandes approach and Dijkstra approach. The representation of the figure is consider the result of Brandes as ground truth. The vertex with the lowest error is *4908*. The nodes with index *1* has the highest error this is happening because it was used as a source node for the calculation of shortest paths using Dijkstra and is bias.

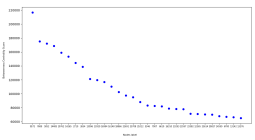

Figure 1: Arxiv Betweenness Centrality Brandes

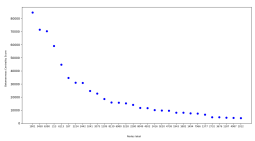

Figure 2: MusaeFacebook Betweenness Centrality Brandes

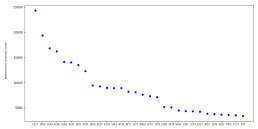

Figure 3: LastfmAsia Betweenness Centrality Brandes

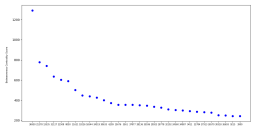

Figure 4: Deezer Betweenness Centrality Brandes

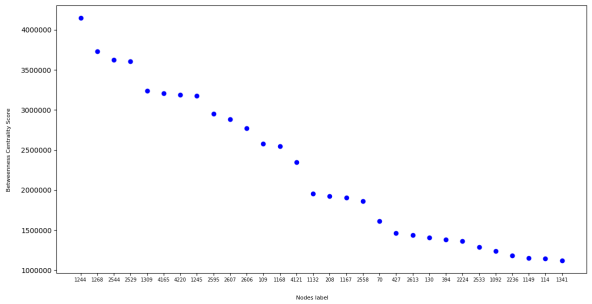

Figure 5: Usgrid Betweenness Centrality Brandes

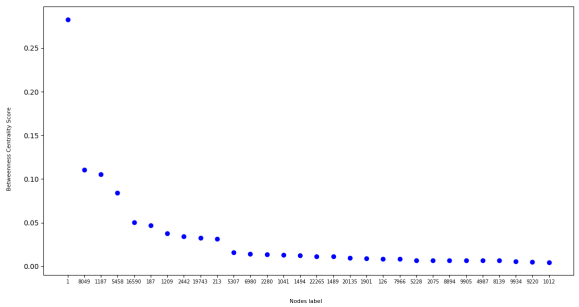

Figure 6: MusaeFacebook Betweenness Centrality DijkstraFreeman

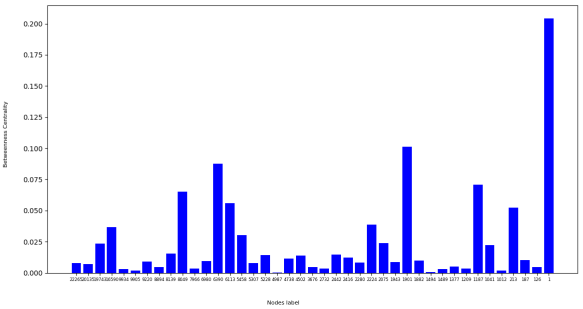

Figure 7: The absolute difference of betweenness centrality Freeman and Brandes

Results			
Dataset	SCC	time Brandes	time Dijkstra
Musae_Facebook	0.9	45 sec	approximately 654800 sec

Table 1

(2) Correlation: We show betweenness centrality score distributions are correlated with each other.
A ranked-based spearman correlation coefficient(SCC) it represents a metric that can calculate and measure how well the relationship of the betweenness centrality of Freeman and Djikstra algorithm 3 and the betweenness centrality of Brandes equation 1 proposed by [7].
SCC depicts the correlation for each vertex betweenness centrality score. It ranks the score of each vertex base on it's score using two different centrality scores. Where $d_i = BCDijkstraFreeman_i - BCbrandes_i$ is the difference ranking of vertex. Before proceed each vertex was sorted in ascending order. The result depicts a final ranking of correlation.

$$SCC(M, C) = 1 - \frac{6\sum_{i=1}^{n} d^2}{n(n^2 - 1)} \quad (9)$$

The coefficient value was bellow 1 and depict a moderate correlation between the betweenness centrality scores. The score depict that it is less unlikely to reject the hypothesis of correlation between the score of Freeman and Brandes table 1.

(3) Computation time: We show that the proposed algorithm performs considerably fast with the small sample size. A scatter plot depicts the error on each graph for measuring the centrality of two given algorithms. Also, we scratch a part of computation time from the paper [8], we can see the comparison result in the figure 10

$$Error_i = |BCbrandes_i - BCDijkstraFreeman_i| \quad (10)$$

A scatter plot 9 depicts computation time for each node.

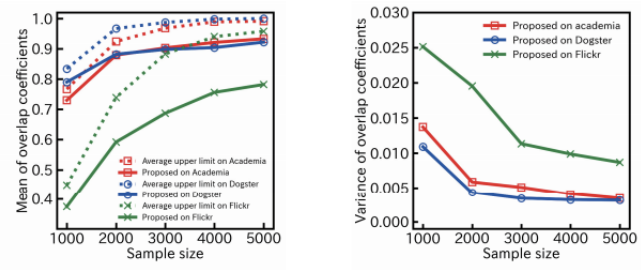

Figure 8: Error

The nodes that needed the most time to calculate their computation time was the the node *1* and the node *8049*.

$$\texttt{Computation time} = \frac{SSP}{n} \quad (11)$$

7. CONCLUSION

Following a calculation based on different calculation equations, it was defined a way to produce the betweenness score of a whole and and of a sample using two different algorithms. While providing the problem definition it was discussed alternative way that can be used. Our experiment

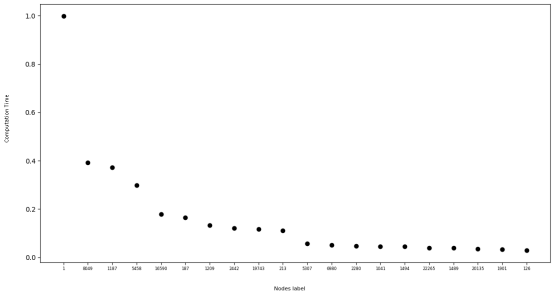

Figure 9: Computation Time Musae Facebook

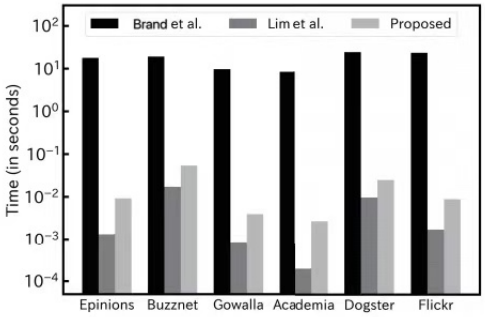

Figure 10: Computation Time

confirmed the feasibility and the utility of our framework and allowed for interesting insights. In particular we showed that two tradition algorithms and a sampling algorithm. The sampling algorithm obtains unbiased estimators of the ego betweenness centrality of sampled nodes via a random walk (around 10-15 % sample rates) and approximates the top-k nodes with the highest betweenness centrality as the top-k nodes with the highest estimated ego betweenness centrality. Given the experimental results, each algorithms has their pros and cons.
In addition from the comparison of the Freeman and Brandes the outcome depicts that the Dijkstra with Freeman need considerable more time than the Brandes to calculate the betweenness centrality. An estimation of Dijkstra and Freeman is that it has some bias result. The correlation coefficient of Spearman depicts moderate degree correlation.
For the real world datasets, we need to tuning algorithms according to the specific problem.

8. REFERENCES

[1] R. Abousleiman and O. Rawashdeh. A bellman-ford approach to energy efficient routing of electric vehicles. In *2015 IEEE Transportation Electrification Conference and Expo (ITEC)*, pages 1–4, 2015.

[2] U. Brandes. A faster algorithm for betweenness centrality. *The Journal of Mathematical Sociology*, 25, 03 2004.

[3] M. Ember. Evidence and science in ethnography: Reflections on the freeman-mead controversy. *American Anthropologist*, 87:906 – 910, 10 2009.

[4] L. C. Freeman. Centrality in social networks conceptual clarification. *Social Networks*, 1(3):215 – 239, 1978.

[5] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, Jun 2002.

[6] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[7] N. Meghanathan. A comprehensive analysis of the correlation between maximal clique size and centrality metrics for complex network graphs. *Egyptian Informatics Journal*, 09 2016.

[8] K. Nakajima and K. Shudo. Estimating high betweenness centrality nodes via random walk in social networks. *Journal of Information Processing*, 28:436–444, 2020.

[9] W. Peng, X. Hu, F. Zhao, and J. Su. A fast algorithm to find all-pairs shortest paths in complex networks. *Procedia Computer Science*, 9:557 – 566, 2012. Proceedings of the International Conference on Computational Science, ICCS 2012.

[10] L. Roditty and U. Zwick. On dynamic shortest paths problems. *Algorithmica*, 61:389–401, 2010.

[11] R. A. Rossi and N. K. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, 2015.