5-28-2010

Path Centrality: A New Centrality Measure in Networks

Tharaka Alahakoon
University of South Florida

Path Centrality: A New Centrality Measure in Networks

by

Tharaka Alahakoon

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Computer Science
Department of Computer Science and Engineering
College of Engineering
University of South Florida

Major Professor: Rahul Tripathi, Ph.D.
Adriana Iamnitchi, Ph.D.
Miguel Labrador, Ph.D.

Date of Approval:
May 28, 2010

Keywords: betweenness centrality, social networks, randomized algorithms, experimental algorithmics, graphs

DEDICATION

to my parents and to my two brothers

ACKNOWLEDGEMENTS

It is my pleasure to thank those who made this thesis possible; this work would not have been completed without help and support of many individuals. I am grateful to everyone who has fostered me with their advice and their suggestions. In particular, I would like to deeply express my sincere gratitude to my advisor Dr. Rahul Tripathi for providing me an opportunity to conduct my Master's thesis under him. I thank him for his patience, motivation, enthusiasm, and immense knowledge to guide and support me over the course of it. I am grateful to him for the long discussions that helped me sort out the technical details of my work and for his careful reading and commenting on countless revisions of this thesis.

I wish to also express my warm and sincere thanks to Dr. Adriana Iamnitchi for many valuable hours of discussions that have helped me understand my experimental area better. I thank her for providing me with very valuable data sets for my experiments and for her guidance in the experimental portion of this thesis. I would like to acknowledge Dr. Miguel Labrador for his comments during my thesis defense, in particular the suggestion of using gnuplots for plotting graphs.

I also like to thank the use of the services provided by Research Computing, University of South Florida. I am grateful to many people on the faculty and staff of the Department of Computer Science and Engineering and the Department of Mathematics and Statistics at the University of South Florida for their assistance and encouragement in various ways during my course of studies. I like to thank many friends and colleagues who have helped me through these difficult years. Their support and care strengthened me to overcome setbacks and stay focused on my graduate study. I greatly value their friendship and I deeply appreciate their belief in me.

Finally, but not the least, I thank my family for believing in me and loving me for all these years. Their love and support kept me moving forward in many joyful as well as difficult times.

TABLE OF CONTENTS

LIST OF TABLES ii

LIST OF FIGURES iv

LIST OF ALGORITHMS vi

ABSTRACT vii

CHAPTER 1 INTRODUCTION 1
- 1.1 Motivation 1
- 1.2 Notations 1

CHAPTER 2 CENTRALITY MEASURES 3
- 2.1 Popular Centrality Measures 3
- 2.2 Betweenness Centrality 5
- 2.3 Bounded-Distance Betweenness Centrality for Unweighted Graphs 10

CHAPTER 3 k-PATH CENTRALITY 12
- 3.1 The Notion of k-Path Centrality for Unweighted Graphs 12
- 3.2 The Notion of k-Path Centrality for Weighted Graphs 15
- 3.3 A Randomized Approximation Algorithm for k-Path Centrality 17

CHAPTER 4 RESULTS 20
- 4.1 Randomly Generated Networks 20
- 4.2 Real Networks 29

CHAPTER 5 CONCLUSION 51

REFERENCES 53

ABOUT THE AUTHOR End Page

LIST OF TABLES

Table 1.1	Notations	2
Table 2.1	Runtime of each procedure in Brandes' betweenness centrality algorithm	9
Table 4.1	Summary information of a random network with 5000 vertices and 10% density	23
Table 4.2	Summary information of a random network with 5000 vertices and 20% density	24
Table 4.3	Summary information of a random network with 5000 vertices and 30% density	25
Table 4.4	Summary information of a random network with 5000 vertices and 40% density	26
Table 4.5	Summary information of a random network with 5000 vertices and 50% density	27
Table 4.6	Summary information of a random network with 5000 vertices and 60% density	28
Table 4.7	Summary information of Zachary's Karate Club data set	33
Table 4.8	Summary information of Les Miserables data set	34
Table 4.9	Summary information of Word Adjacencies data set	35
Table 4.10	Summary information of Condensed Matter 1999 data set	36
Table 4.11	Summary information of Condensed Matter 2003 data set	37
Table 4.12	Summary information of Condensed Matter 2005 data set	38
Table 4.13	Summary information of High-Energy Theory data set	39
Table 4.14	Summary information of Internet data set	40
Table 4.15	Summary information of Yeast data set	41
Table 4.16	Summary information of Kazaa data set 1	42
Table 4.17	Summary information of Kazaa data set 2	43
Table 4.18	Summary information of Kazaa data set 3	44

Table 4.19 Summary information of Computational Geometry data set 45

Table 4.20 Summary information of Small World data set 46

Table 4.21 Summary information of Small, Griffith and Descendants data set 47

Table 4.22 Summary information of Scientometrics data set 48

Table 4.23 Summary information of Self-Organizing Maps data set 49

Table 4.24 Summary information of Zewail data set 50

LIST OF FIGURES

Figure 4.1	Plot showing running times (in seconds) of Brandes' algorithm and our randomized approximation algorithm	21
Figure 4.2	Scatter plot for a random network with 5000 vertices and 10% density	23
Figure 4.3	Scatter plot for a random network with 5000 vertices and 20% density	24
Figure 4.4	Scatter plot for a random network with 5000 vertices and 30% density	25
Figure 4.5	Scatter plot for a random network with 5000 vertices and 40% density	26
Figure 4.6	Scatter plot for a random network with 5000 vertices and 50% density	27
Figure 4.7	Scatter plot for a random network with 5000 vertices and 60% density	28
Figure 4.8	Scatter plot for Zachary's Karate Club data set	33
Figure 4.9	Scatter plot for Les Miserables data set	34
Figure 4.10	Scatter plot for Word Adjacencies data set	35
Figure 4.11	Scatter plot for Condensed Matter 1999 data set	36
Figure 4.12	Scatter plot for Condensed Matter 2003 data set	37
Figure 4.13	Scatter plot for Condensed Matter 2005 data set	38
Figure 4.14	Scatter plot for High-Energy Theory data set	39
Figure 4.15	Scatter plot for Internet data set	40
Figure 4.16	Scatter plot for Yeast data set	41
Figure 4.17	Scatter plot for Kazaa data set 1	42
Figure 4.18	Scatter plot for Kazaa data set 2	43
Figure 4.19	Scatter plot for Kazaa data set 3	44
Figure 4.20	Scatter plot for Computational Geometry data set	45
Figure 4.21	Scatter plot for Small World data set	46
Figure 4.22	Scatter plot for Small, Griffith and Descendants data set	47
Figure 4.23	Scatter plot for Scientometrics data set	48

Figure 4.24 Scatter plot for Self-Organizing Maps data set 49

Figure 4.25 Scatter plot for Zewail data set 50

LIST OF ALGORITHMS

1	Initialize	7
2	Relax&Count	8
3	Accumulation	8
4	Brandes' Betweenness	9
5	Bounded-Distance Betweenness	11
6	Indicator	13
7	Unweighted k-Path	14
8	Weighted k-Path	17
9	Randomized Approximation k-Path	18

Path Centrality: A New Centrality Measure in Networks

Tharaka Alahakoon

ABSTRACT

In network analysis, it is useful to identify important vertices in a network. Based on the varying notions of importance of vertices, a number of centrality measures are defined and studied in the literature. Some popular centrality measures, such as betweenness centrality, are computationally prohibitive for large-scale networks. In this thesis, we propose a new centrality measure called k-path centrality and experimentally compare this measure with betweenness centrality.

We present a polynomial-time randomized algorithm for distinguishing high k-path centrality vertices from low k-path centrality vertices in any given (unweighted or weighted) graph. Specifically, for any graph $G = (V, E)$ with n vertices and for every choice of parameters $\alpha \in (0, 1)$, $\epsilon \in (0, 1/2)$, and integer $k \in [1, n]$, with probability at least $1 - 1/n^2$ our randomized algorithm distinguishes all vertices $v \in V$ that have k-path centrality $\mathcal{C}_k(v)$ more than $n^\alpha(1 + 2\epsilon)$ from all vertices $v \in V$ that have k-path centrality $\mathcal{C}_k(v)$ less than $n^\alpha(1 - 2\epsilon)$. The running time of the algorithm is $\mathcal{O}(k^2\epsilon^{-2}n^{1-\alpha} \ln n)$.

Next, we present a polynomial-time randomized approximation algorithm for computing the k-path centrality values of all vertices in any given (unweighted or weighted) graph. Specifically, for any graph and for every choice of parameters $\alpha \in (0, 1/2)$ and integer $k \in [1, n]$, with probability at least $1 - 1/n^2$ our randomized approximation algorithm computes the k-path centrality value of every vertex within an additive error of at most $n^{1/2+\alpha}$. The running time of the algorithm is $\mathcal{O}(k^3 n^{1-2\alpha} \ln n)$.

Theoretically and experimentally, our algorithms are (for suitable choices of parameters) significantly faster than the best known deterministic algorithm for computing exact betweenness centrality values (Brandes' algorithm). Through experimentations on both real and ran-

domly generated networks, we demonstrate that vertices that have high betweenness centrality values also have high k-path centrality values.

CHAPTER 1

INTRODUCTION

1.1 Motivation

A network is a (directed or undirected) graph in which vertices typically represent people or organizations or entities, and edges denote relationships or likeliness or physical connections. A fundamental problem in the analysis of networks is to determine vertices or edges of great importance in a given network. The motivation of this thesis is to find important vertices of a network in significantly fast time.

There are many centrality measures proposed to evaluate the importance of a vertex or an edge in a network. Betweenness centrality is considered to be the most popular of all. The best known deterministic algorithm for computing betweenness centrality of all vertices (or edges) takes $\mathcal{O}(n+m)$ space and $\mathcal{O}(nm)$ time in unweighted graphs, and $\mathcal{O}(n+m)$ space and $\mathcal{O}(nm + n^2 \log n)$ time in weighted graphs [Bra01], where n is the number of vertices and m is the number of edges in the graph. In this thesis, we introduce a new centrality measure called k-path centrality and devise a polynomial-time randomized approximation algorithm for computing all k-path centrality values with small error probability. Our randomized approximation algorithm is significantly faster than that for computing betweenness centrality of all vertices. We also experimentally demonstrate that vertices with high betweenness centrality value also have high k-path centrality value. Therefore, we believe that the k-path centrality notion and the randomized approximation algorithm proposed in this thesis may have practical merit in network analysis.

1.2 Notations

In a network, it is critical for various practical reasons to identify important vertices or edges. In this thesis, we will focus on importance of vertices, but the concept can also be

applied for importance of edges. For convenience, we can describe a network as a (directed or undirected) graph $G = (V, E)$, where V represents the set of vertices and E represents the set of edges. Let n and m denote the cardinality of V and the cardinality of E, respectively. Let W be a nonnegative weight function on the edges of a graph. Without loss of generality, we can assume that the graphs considered in this thesis are edge-weighted, since the edges of an unweighted graph can be assumed to have unit weights. Also, we can assume all graphs considered in this thesis to be directed, since any undirected edge $\{u, v\}$ can be split into two directed edges (u, v) and (v, u).

Let $G = (V, E)$ be a graph and let s, t be arbitrary vertices of G. We define a *path* from s to t as a sequence of vertices that starts from s and ends at t such that any two consecutive vertices in the sequence are connected by an edge of G. The *weight* of a path is defined to be the sum of the weights of all the edges in the path. The *shortest path* from s to t in G is a path of minimum weight, and we denote its weight by $d_G(s, t)$. Let $P_s(t)$ be the *set of predecessors* of a vertex t on shortest paths from s to t in G. Let σ_{st} denote the *number of shortest paths* from s to t in G. For any $v \in V$, let $\sigma_{st}(v)$ denote the number of shortest paths from s to t in G that pass through v. Note that $d_G(s, s) = 0$, $\sigma_{ss} = 1$ and $\sigma_{st}(v) = 0$ if $v \in \{s, t\}$.

Some useful notations used in this thesis are summarized in Table 1.1.

Table 1.1 Notations

Notation	Meaning
V	Set of vertices in graph $G = (V, E)$
E	Set of edges in graph $G = (V, E)$
n	Cardinality of V
m	Cardinality of E
$W(u, v)$	Weight of the directed edge from $u \in V$ to $v \in V$, where $(u, v) \in E$
$d_G(s, t)$	The weight of a shortest path from $s \in V$ to $t \in V$
$P_s(t)$	Set of predecessors of a vertex t on shortest paths from $s \in V$ to $t \in V$
σ_{st}	The number of shortest paths from $s \in V$ to $t \in V$
$\sigma_{st}(v)$	The number of shortest paths from $s \in V$ to $t \in V$ that pass through $v \in V$

CHAPTER 2

CENTRALITY MEASURES

2.1 Popular Centrality Measures

A major focus in network analysis is on finding important vertices or edges in a given network. Centrality measures are used to rank vertices (or edges) based on their importance in the network. Henceforward, we will consider vertex-based centrality measures and ignore their edge-based counterparts, since the latter are defined analogously to the former. Some of the earliest studies of network analysis occurred in the field of social science [WF94, Sco00]. Over the years, network researchers have introduced many indices to measure the centrality of a vertex. Simplest centrality measure is the degree. Because degree of a vertex only depends on its neighbors, the degree measure does not give a good indication of the vertex compared to the whole network. Therefore centrality measures, which indicate the rank of a vertex according to its position in the network, were more preferable. Some popular centrality measures are Closeness [Bea65, Sab66], Eigenvector [Bon72], Graph [HH95], Stress [Shi53] and Betweenness [Ant71, Fre77].

Definition 2.1 (Closeness Centrality [Bea65]) *For every vertex $v \in V$ of a directed weighted graph $G(V, E)$, the* closeness centrality $\mathcal{C}_C(v)$ *of v is defined by*

$$\mathcal{C}_C(v) = \frac{n-1}{\sum_{t \in V} d_G(v,t)}.$$

Definition 2.2 (Graph Centrality [HH95]) *For every vertex $v \in V$ of a directed weighted graph $G(V, E)$, the* graph centrality $\mathcal{C}_G(v)$ *of v is defined by*

$$\mathcal{C}_G(v) = \frac{1}{\max_{t \in V} d_G(v,t)}.$$

Definition 2.3 (Stress Centrality [Shi53]) *For every vertex $v \in V$ of a directed weighted graph $G(V, E)$, the* stress centrality $\mathcal{C}_S(v)$ *of v is defined by*

$$\mathcal{C}_S(v) = \sum_{s \neq v} \sum_{t \neq v,s} \sigma_{st}(v).$$

Definition 2.4 (Betweenness Centrality [Ant71, Fre77]) *For every vertex $v \in V$ of a directed weighted graph $G(V, E)$, the* betweenness centrality $\mathcal{C}_B(v)$ *of v is defined by*

$$\mathcal{C}_B(v) = \sum_{s \neq v} \sum_{t \neq v,s} \frac{\sigma_{st}(v)}{\sigma_{st}}.$$

According to above definitions, high centrality score of a vertex indicates that the vertex is easily reachable from other vertices. As an example, suppose that we want to know who are the most important people in a research group. If we assign edge weights between two people according to the number of papers they published together, we get an undirected, weighted graph. As mentioned before, we can treat this graph as a directed graph by splitting each edge into bidirectional edges. Any of the above centrality measure will assign each vertex a value. By arranging these values in a decreasing order we can figure out who are the most important researchers in that group. Note that we might get different ordering depending on the centrality measure used. It is arguable to say which centrality measure best describes the importance of vertices in a graph. Centrality measures assign values to vertices that allow us to compare two or more vertices. Typically, the higher the centrality value, the greater the importance of that vertex.

The computation of these centrality measures is a time consuming task. A lot of research is devoted to devising fast algorithms for computing these measures. Betweenness centrality measure captures the degree of influence a vertex has over the information flow in a network. Therefore, in recent years betweenness centrality has gained popularity among many social network analysts. The best known exact algorithm for computing betweenness centrality is given by Brandes [Bra01]. Computation of the betweenness centrality using Brandes' algorithm takes $\mathcal{O}(n + m)$ space and $\mathcal{O}(nm)$ time on unweighted graphs, and $\mathcal{O}(n + m)$ space and $\mathcal{O}(nm + n^2 \log n)$ time on weighted graphs, where n is the number of vertices and m is

the number of edges. Based on the idea used by Eppstein and Wang [EW01], Brandes and Pich [BP07] proposed an approximation algorithm to compute betweenness centrality. Their algorithm randomly chooses a small sample of vertices called pivots. It then computes single source shortest paths from each pivot to all other vertices. Then the algorithm defines the estimated betweenness centrality of a vertex in terms of the average contribution of each pivot. The accuracy and the runtime of the algorithm depend on the number of pivots selected.

Geisberger, Sanders and Schultes [GSS08] proposed linear scaling and bisection scaling techniques to approximate betweenness centrality. In linear scaling they assumed that the contribution of the sample depends linearly on the distance to the sample and can be implemented using a slight variation of the Brandes' betweenness algorithm. In bisection scaling contribution of the sample is only on the second half of the path and requires a different approach with another level of random sampling. For a given vertex, Bader et al. [BKMM07] presented an adaptive sampling based algorithm for approximately computing betweenness centrality of a vertex. They showed that if the betweenness centrality of a vertex is n^2/t, for some constant $t \geq 1$, then with probability at least $1 - 2\epsilon$, its betweenness centrality can be estimated to within a factor of $1/\epsilon$, where $0 < \epsilon < 1/2$, using ϵt samples of source vertices.

We next visit betweenness centrality in detail.

2.2 Betweenness Centrality

Anthonisse [Ant71] and Freeman [Fre77] independently introduced the betweenness centrality of a vertex. The computation of betweenness centrality of a vertex $v \in V$ requires two steps:

- counting the number of shortest paths between all pairs (s, t) of vertices of G
- counting the number of such paths between s, t that pass through v.

Note that a vertex v lies on a shortest path from s to t if and only if the equality $d_G(s, t) = d_G(s, v) + d_G(v, t)$ is satisfied. We can compute the weight and the number of shortest paths between any pair (s, t) of vertices using traversal algorithms such as the breadth-first search algorithm for unweighted graphs and Dijkstra's algorithm for weighted graphs. For computing

σ_{st}, we use the fact that $\sigma_{st} = \sum_{v \in P_s(t)} \sigma_{sv}$, where $P_s(t)$ denotes the set of predecessors of the vertex t on shortest paths from s to t. After computing σ_{st} values for all $s, t \in V$ we can compute $\sigma_{st}(v)$ using the following fact:

$$\sigma_{st}(v) = \begin{cases} \sigma_{sv} \cdot \sigma_{vt} & \text{if } d_G(s,t) = d_G(s,v) + d_G(v,t), \\ 0 & \text{otherwise.} \end{cases}$$

This simple algorithm for computing betweenness centrality requires $\mathcal{O}(n^2)$ space and $\mathcal{O}(n^3)$ time. Calculating betweenness centrality in networks with huge number of vertices is computationally prohibitive. Brandes [Bra01] was able to compute betweenness centrality in $\mathcal{O}(n+m)$ space and $\mathcal{O}(nm)$ time on unweighted graphs, and $\mathcal{O}(n+m)$ space and $\mathcal{O}(nm + n^2 \log n)$ time on weighted graphs. Brandes defined the notation of the *dependency* of a vertex $s \in V$ on a single vertex $v \in V$ as $\delta_{s\star}(v) = \sum_{t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$ and proved the following recursive relation on $\delta_{s\star}(v)$, which is significant to his algorithm:

$$\delta_{s\star}(v) = \sum_{w : v \in P_s(w)} \frac{\sigma_{sv}}{\sigma_{sw}} (1 + \delta_{s\star}(w)).$$

Below we give Brandes' betweenness centrality algorithm for weighted graphs. Unweighted case can be done much simpler using breadth-first search instead of Dijkstra's algorithm. The pseudocode for Brandes' betweenness algorithm is described in the procedure Brandes' Betweenness. This procedure makes calls to the procedures Initialize, Relax&Count, and Accumulation.

The procedure Initialize takes a graph $G(V, E)$, array d of distances, array σ of number of shortest paths, array P of predecessors, queue Q, stack S, and a start vertex s as input. It does the following:

- For every vertex v except s, the distance $d[v]$ from s to v is set to ∞. The distance $d[s]$ from s to itself is set to 0.
- For every vertex v except s, the number of shortest paths $\sigma[v]$ from s to v is set to 0. The value $\sigma[s]$ is set to 1.

- For every vertex v, the predecessor $P[v]$ of v is set to $\emptyset$.
- Add vertex s to the queue Q.
- Set stack S to $\emptyset$.

input : Graph $G = (V, E)$, Array d, Array σ, Array P, Queue Q, Stack S, Vertex s

output: None

begin

  foreach (vertex $v \in V$) **do**

    $d[v] \leftarrow \infty$;

    $\sigma[v] \leftarrow 0$;

    $P[v] \leftarrow \emptyset$;

  $d[s] \leftarrow 0$;

  $\sigma[s] \leftarrow 1$;

  $Q \leftarrow \{s\}$;

  $S \leftarrow \emptyset$;

end

Algorithm 1: Initialize

The procedure Relax&Count takes two vertices u and v of a graph G, array d of distance, array σ of number of shortest paths, array P of predecessors, queue Q, and array W of edge weights as input. It does the following:

- If the current distance estimate $d[v]$ of v is greater than the sum of the distance estimate $d[u]$ of u and the weight $W(u, v)$ of edge (u, v), then
  - set $d[v]$ to the latter value.
  - set $\sigma[v] \leftarrow \sigma[u]$.
  - set u as the only predecessor of v.
  - insert/update v to the queue Q with new key $d[v]$.
- If the current distance estimate $d[v]$ of v is equal to the sum of the distance estimate $d[u]$ of u and the weight $W(u, v)$ of edge (u, v), then

– set $\sigma[v] \leftarrow \sigma[v] + \sigma[u]$.

– append u to the predecessor list of v.

input : Array d, Array σ, Array P, Queue Q, Array W, Vertex u, Vertex v
output: None

begin
  if $(d[v] > d[u] + W(u, v))$ **then**
    $d[v] \leftarrow d[u] + W(u, v)$;
    $\sigma[v] \leftarrow \sigma[u]$;
    $P[v] \leftarrow \{u\}$;
    insert/update Q by setting $Q \leftarrow v$ with new key $d[v]$;
  if $(d[v] = d[u] + W(u, v))$ **then**
    $\sigma[v] \leftarrow \sigma[v] + \sigma[u]$;
    $P[v] \leftarrow P[v] \cup \{u\}$;
end

Algorithm 2: Relax&Count

The procedure Accumulation takes a graph G, array $\mathcal{C}_B$ of betweenness centrality measures, array σ of number of shortest paths, array P of predecessors, stack S, and a start vertex s as input. It does the following:

- For every vertex v of G, the dependency of the vertex s on v that contributes to the betweenness centrality $\mathcal{C}_B(v)$ of v is computed.

input : Graph $G = (V, E)$, Array $\mathcal{C}_B$, Array σ, Array P, Stack S, Vertex s
output: None

begin
  foreach (vertex $v \in V$) **do**
    $\delta[v] \leftarrow 0$;
  while (S is nonempty) **do**
    pop $w \leftarrow S$;
    for $(v \in P[w])$ **do**
      $\delta[v] \leftarrow \delta[v] + \frac{\sigma[v]}{\sigma[w]} \cdot (1 + \delta[w])$;
    if $(w \neq s)$ **then**
      $\mathcal{C}_B(w) \leftarrow \mathcal{C}_B(w) + \delta[w]$;
end

Algorithm 3: Accumulation

The procedure Brandes' Betweenness takes a graph G and array W of edge weights as input. It does the following:

- For every vertex v, its betweenness centrality $\mathcal{C}_B(v)$ is computed.

```
input  : Graph G = (V, E), Array W
output: Array C_B
begin
    foreach (vertex v ∈ V) do
        C_B(v) ← 0 ;
    foreach (vertex s ∈ V) do
        Initialize(G, d, σ, P, Q, S, s) ;
        while (Q is nonempty) do
            u ← Extract-Min(Q) ;
            push u to S ;
            foreach (vertex v such that (u, v) ∈ E) do
                Relax&Count(d, σ, P, Q, W, u, v) ;
        Accumulation(G, C_B, σ, P, S, s) ;
    return C_B ;
end
```

Algorithm 4: Brandes' Betweenness

Analysis: Brandes' betweenness centrality algorithm consists of several procedures. Based on a *fibonacci heap* implementation of the priority queue Q, the running time of these procedures can be summarized as follows.

Table 2.1 Runtime of each procedure in Brandes' betweenness centrality algorithm

Procedure	Runtime
Initialize	$\mathcal{O}(n)$
Relax&Count	amortized $\mathcal{O}(1)$
Accumulation	$\mathcal{O}(n)$
Brandes' Betweenness	$\mathcal{O}(nm + n^2 \log n)$

Thus, the overall running time of Brandes' betweenness centrality algorithm is $\mathcal{O}(nm + n^2 \log n)$. Using breadth-first search instead of Dijkstra's algorithm when the input graph is unweighted, the computation time of Brandes' betweenness centrality algorithm reduces to $\mathcal{O}(nm)$ time.

2.3 Bounded-Distance Betweenness Centrality for Unweighted Graphs

In betweenness centrality for unweighted graphs, we measure shortest paths irrespective of their length. Borgatti and Everett [BE06] gave the idea of limiting the length. Their argument was that very long paths are seldom used. They called this measure k-betweenness centrality (because they only consider depth at most k). Later Brandes [Bra08] redefined this measure as bounded-distance betweenness centrality.

Definition 2.5 (Bounded-Distance Betweenness Centrality [Bra08]) *For any fixed $k \in \mathbb{N}^+$ and for every vertex $v \in V$ of a directed unit-weighted (i.e., unweighted) graph $G = (V, E)$, the* bounded-distance betweenness centrality $\mathcal{C}_{B(k)}(v)$ *of v is defined by*

$$\mathcal{C}_{B(k)}(v) = \sum_{s,t \in V : d_G(s,t) \leq k} \frac{\sigma_{st}(v)}{\sigma_{st}}.$$

Bounded-distance betweenness centrality for an unweighted graph can be explicitly computed using Brandes' betweenness centrality algorithm where we stop the breadth-first search of the algorithm when a vertex of distance k is reached [Bra08].

The procedure Bounded-Distance Betweenness takes a graph G, integer $k \in \mathbb{N}^+$, and an array W representing the weight function on the edges of G as input. It does the following:

- For every vertex v, the bounded-distance betweenness $\mathcal{C}_{B(k)}(v)$ is computed.

```
input  : Graph G, Integer k ∈ ℕ⁺, Array W such that W(e) = 1 for every edge e
output: Array C_{B(k)}

begin
    foreach (vertex s ∈ V) do
        C_{B(k)}(u) ← 0 ;
    foreach (vertex s ∈ V) do
        Initialize(G, d, σ, P, Q, S, s) ;
        while (Q is nonempty) do
            u ← Extract-Min(Q) ;
            if (d[u] > k) then
                break ;
            push u to S ;
            foreach (vertex v such that (u, v) ∈ E) do
                Relax&Count(d, σ, P, Q, W, u, v) ;
        Accumulation(G, C_{B(k)}, σ, P, S, s) ;
    return C_{B(k)} ;
end
```

Algorithm 5: Bounded-Distance Betweenness

Analysis: Bounded-distance betweenness algorithm above is a restricted version of Brandes' betweenness algorithm. Since traversal of the graph is done from each vertex to other vertices that are at distance at most k, the while loop breaks after reaching the distance k. In the worst case, if distances from any vertex to all other vertices are less than k, this algorithm will perform similar to Brandes' betweenness algorithm, which takes $\mathcal{O}(nm)$ time on unweighted graphs.

CHAPTER 3

k-PATH CENTRALITY

3.1 The Notion of k-Path Centrality for Unweighted Graphs

In the previous chapter, we introduced some popular centrality measures. Roughly speaking, betweenness centrality of a vertex determines how important that vertex is in relation to the overall information flow in the network. Therefore, it is considered to be an important centrality measure for social networks. Brandes' betweenness centrality algorithm [Bra01] is the best known deterministic algorithm for computing betweenness of all the vertices of a graph. However, a significant shortcoming of Brandes' algorithm is that its running time is prohibitive for large-scale networks. With the motivation to find the most important vertices in a network, in this chapter we introduce a new centrality measure called k-path centrality.

Since unweighted case is easier to grasp, first we will define k-path centrality for unweighted graphs. For any $s \in V$, let $p(s;\ell)$ denote the number of paths of length exactly $\ell \in \mathbb{N}^+$ that start from s. For any $s, v \in V$, let $p_v(s;\ell)$ denote the number of paths of length exactly $\ell \in \mathbb{N}^+$ that start from s and pass through v.

Definition 3.1 (k-Path Centrality for Unweighted Graphs) *For any fixed $k \in \mathbb{N}^+$ and for every vertex $v \in V$ of a directed unit-weighted (i.e., unweighted) graph $G(V, E)$, the* k-path centrality $\mathcal{C}_k(v)$ *of v is defined by*

$$\mathcal{C}_k(v) = \sum_{s \neq v} \sum_{1 \leq \ell \leq k} \frac{p_v(s;\ell)}{p(s;\ell)}.$$

We present a randomized algorithm with small error probability (at most $1/n^2$), that distinguishes vertices with high k-path centrality values from vertices with low k-path centrality values, where n is the number of vertices in the graphs. Specifically, for every choice of real numbers $\alpha \in (0,1), \epsilon \in (0, 1/2)$ and fixed integer $k \in [1, n]$, with probability at least $1 - 1/n^2$

if a vertex $v \in V$ has k-path centrality $\mathcal{C}_k(v) > n^{\alpha}(1+2\epsilon)$, then our randomized algorithm reports that v is a vertex with high k-path centrality, and if a vertex $v \in V$ has k-path centrality $\mathcal{C}_k(v) < n^{\alpha}(1-2\epsilon)$, then our randomized algorithm reports that v is a vertex with low k-path centrality. The running time of our algorithm for every choice of random sequence is $\mathcal{O}(k^2\epsilon^{-2}n^{1-\alpha}\ln n)$. Hence, for small integers $k = O(n^{1/2})$ and for fixed real numbers $\alpha \in (0,1)$ and $\epsilon \in (0,1/2)$, the running time of our algorithm is significantly less than that of Brandes' algorithm that runs in time $\mathcal{O}(mn)$ on unweighted graphs.

We next present our algorithm (Algorithm 7) for k-path centrality in unweighted graphs. This algorithm uses a procedure, called Indicator. The procedure Indicator takes a graph G, a real number $\epsilon \in (0,1/2)$, and an array B as input. It does the following:

- For every vertex v, if $B(v) > 6\ln(n)/\epsilon^2$ then vertex v is reported as a *high* k-path centrality vertex. Otherwise it is reported as a *low* k-path centrality vertex.

```
input  : Graph G, real ε ∈ (0, 1/2), Array B
output: None
begin
    n ← |V(G)| ;
    foreach (vertex v ∈ V) do
        if B(v) > 6 ln(n)/ε² then
            Print "v has a high k-path centrality" ;
        else
            Print "v has a low k-path centrality" ;
end
```

Algorithm 6: Indicator

The procedure Unweighted k-Path takes an unweighted graph G, real value $\alpha \in (0,1)$, real value $\epsilon \in (0,1/2)$, and integer $k \in [1,n]$ as input. It does the following:

- For every vertex v, indicate whether it is a *high* k-path centrality vertex or a *low* k-path centrality vertex.

input : Graph G, Real $\alpha \in (0,1)$, Real $\epsilon \in (0,1/2)$, Integer $k \in [1,n]$

output: None

begin

  foreach (vertex $v \in V$) **do**
    $B(v) \leftarrow 0$;
    $v \leftarrow$ unexplored ;
  $n \leftarrow |V(G)|$;
  Stack $S \leftarrow \emptyset$;
  outer-loop $\leftarrow 1$;
  while (outer-loop $\leq 6k\epsilon^{-2}n^{1-\alpha}\ln n$) **do**
    Let $k' \in [1,k]$ be an integer chosen uniformly at random ;
    Let $x \in V$ be a vertex chosen uniformly at random;
    $x \leftarrow$ explored ;
    push x to S ;
    inner-loop $\leftarrow 1$;
    while (inner-loop $\leq k'$ and not all outgoing edges from x are explored) **do**
      Let (x,y) be an edge from x to an unexplored vertex y, chosen uniformly at random ;
      $y \leftarrow$ explored ;
      push y to S ;
      $B(y) \leftarrow B(y) + 1$;
      inner-loop $\leftarrow$ inner-loop $+1$;
      $x \leftarrow y$;
    while (S is nonempty) **do**
      pop $x \leftarrow S$;
      $x \leftarrow$ unexplored ;
    outer-loop $\leftarrow$ outer-loop $+1$;
  Indicator(G, ϵ, B) ;

end

Algorithm 7: Unweighted k-Path

Analysis: Let $\ell = 6k\epsilon^{-2}n^{1-\alpha}\ln n$. Let vertex v be a high k-path centrality vertex. Then $\mathcal{C}_k(v) > n^{\alpha}(1+2\epsilon)$. Define, for $1 \leq i \leq l$,

$$X_i = \begin{cases} 1 & \text{if the } i^{th} \text{ random walk passes through } v, \\ 0 & \text{otherwise.} \end{cases}$$

It is easy to see that when the algorithm terminates $B(v) = \sum_{i=1}^{\ell} X_i$. Note that

$$\begin{aligned} E[X_i] &= \Pr(X_i = 1) \\ &= \frac{1}{kn}\sum_{s\in V}\sum_{1\leq \ell \leq k} \Pr(\text{a random walk of length } \ell \text{ starts from } s \text{ and passes through } v) \\ &= \frac{1}{kn}\cdot 0 + \frac{1}{kn}\sum_{s\neq v}\sum_{1\leq \ell\leq k}\frac{p_v(s;\ell)}{p(s;\ell)} \\ &= \frac{1}{kn}\mathcal{C}_k(v). \end{aligned}$$

Therefore,

$$\begin{aligned} E[B(v)] = \sum_{i=1}^{\ell} E[X_i] &= \frac{\ell}{kn}\mathcal{C}_k(v) \geq \frac{\ell}{kn^{1-\alpha}}(1+2\epsilon) \\ \Pr(B(v) \leq \frac{\ell}{kn^{1-\alpha}}) &\leq \Pr(B(v) \leq (1-\epsilon)E[B(v)]) \\ &\leq e^{\frac{-\epsilon^2}{2}\cdot\frac{\ell}{kn^{1-\alpha}}(1+2\epsilon)} \qquad \text{(by Chernoff bounds)} \\ &\leq e^{-3\ln n} \\ &= n^{-3}. \end{aligned}$$

From the above analysis, we can conclude that the k-path centrality randomized algorithm has an error probability at most n^{-2} and has runtime $\mathcal{O}(\ell k) = \mathcal{O}(k^2\epsilon^{-2}n^{1-\alpha}\ln n)$. ∎

3.2 The Notion of k-Path Centrality for Weighted Graphs

Now we will generalize the k-path centrality notion and define it for weighted graphs. For any $s \in V$ and $\ell \in \mathbb{N}^+$, let $B(s;\ell)$ denote the number of paths π_{st} from s to some vertex $t \in V$ with one of the following properties:

- Total weight of the path π_{st} from s to t is ℓ, i.e., $W(\pi_{st}) = \ell$.
- Total weight of the path π_{st} from s to t is less than ℓ and there is an edge (t, w) that does not belong to π_{st} such that $W(\pi_{st}) + W(t, w) > \ell$.

Similarly, for any $s, v \in V$ and $\ell \in \mathbb{N}^+$, let $B_v(s; \ell)$ denote the number of paths $\pi_{st}(v)$ from s to some vertex $t \in V$ that pass through v and have one of the following properties:

- Total weight of the path $\pi_{st}(v)$ from s to t that passes through v is ℓ, i.e., $W(\pi_{st}(v)) = \ell$.
- Total weight of the path $\pi_{st}(v)$ from s to t that passes through v is less than ℓ and there is an edge (t, w) that does not belong to $\pi_{st}(v)$ such that $W(\pi_{st}(v)) + W(t, w) > \ell$.

Now we can define k-path centrality for weighted case similar to the unweighted case.

Definition 3.2 (k-Path Centrality for Weighted Graphs) *For any fixed $k \in \mathbb{N}^+$ and for every vertex $v \in V$ of a directed weighted graph $G(V, E)$, the* k-path centrality $\mathcal{C}_k(v)$ *of v is defined by*

$$\mathcal{C}_k(v) = \sum_{s \neq v} \sum_{1 \leq \ell \leq k} \frac{B_v(s; \ell)}{B(s; \ell)}.$$

Note that the above definition works for both weighted and unweighted graphs. We give a randomized algorithm (Algorithm 8) that for every choice of real numbers $\alpha \in (0, 1), \epsilon \in (0, 1/2)$ and fixed integer $k \in [1, n]$, with probability at least $1 - 1/n^2$, if a vertex $v \in V$ has k-path centrality $\mathcal{C}_k(v) > n^{\alpha}(1+2\epsilon)$, then the randomized algorithm reports that v is a vertex with high k-path centrality and if a vertex $v \in V$ has k-path centrality $\mathcal{C}_k(v) < n^{\alpha}(1-2\epsilon)$, then the randomized algorithm reports that v is a vertex with low k-path centrality. We show that the running time of this algorithm is $\mathcal{O}(k^2\epsilon^{-2}n^{1-\alpha}\ln n)$. Here we use the same Indicator procedure as in the unweighted case, but the Weighted k-Path procedure will be little different.

The procedure Weighted k-Path takes a graph G, real value $\alpha \in (0, 1)$, real value $\epsilon \in (0, 1/2)$, integer $k \in [1, n]$, and array W of edge weights as input. It does the following:

- For every vertex v, indicate whether it is a *high* k-path centrality vertex or a *low* k-path centrality vertex.

The analysis for the weighted case is identical to the unweighted case.

```
input  : Graph G, Array W, Real α ∈ (0,1), Real ε ∈ (0,1/2), Integer k ∈ [1,n]
output: None
begin
    foreach (vertex v ∈ V) do
        B(v) ← 0 ;
        v ← unexplored ;
    n ← |V(G)| ;
    Stack S ← ∅ ;
    outer-loop ← 1 ;
    while (outer-loop ≤ 6kε^{-2} n^{1-α} ln n) do
        Let k' ∈ [1,k] be an integer chosen uniformly at random ;
        Let x ∈ V be a vertex chosen uniformly at random;
        x ← explored ;
        push x to S ;
        inner-loop ← 1 ;
        while (inner-loop ≤ k' and not all outgoing edges from x are explored) do
            Let (x,y) be a random edge from x to an unexplored vertex y, chosen
            with probability inversely proportional to its edge weight ;
            y ← explored ;
            push y to S ;
            B(y) ← B(y) + 1 ;
            inner-loop ← inner-loop +1 ;
            x ← y ;
        while (S is nonempty) do
            pop x ← S ;
            x ← unexplored ;
        outer-loop ← outer-loop +1 ;
    Indicator(G, ε, B) ;
end
```

Algorithm 8: Weighted k-Path

3.3 A Randomized Approximation Algorithm for k-Path Centrality

Our algorithm for approximate computation of k-path centrality values in a network is identical to the Weighted k-Path procedure; the main difference is in the number of iterations. By analysis of the algorithm given below we found the required number of iterations to be $2k^2 n^{1-2\alpha} \ln n$ for every choice of integer $k \in [1, n]$ and real number $\alpha \in (0, 1/2)$.

The procedure Randomized Approximation k-Path (Algorithm 9) takes a graph G, real value α, integer k, and array W of edge weights as input. It does the following:

- For every vertex v, an approximation to the k-path centrality $\mathcal{C}_k(v)$ is computed.

input : Graph G, Array W, Real $\alpha \in (0, 1/2)$, Integer $k \in [1, n]$
output: An approximation to the k-path centrality $\mathcal{C}_k(v)$ is computed

begin
  foreach (vertex $v \in V$) **do**
    $B(v) \leftarrow 0$;
    $v \leftarrow$ unexplored ;
  $n \leftarrow |V(G)|$;
  $\ell \leftarrow 2k^2 n^{1-2\alpha} \ln n$;
  Stack $S \leftarrow \emptyset$;
  outer-loop $\leftarrow 1$;
  while (outer-loop $\leq \ell$) **do**
    Let $k' \in [1, k]$ be an integer chosen uniformly at random ;
    Let $x \in V$ be a vertex chosen uniformly at random;
    $x \leftarrow$ explored ;
    push x to S ;
    inner-loop $\leftarrow 1$;
    while (inner-loop $\leq k'$ and not all outgoing edges from x are explored) **do**
      Let (x, y) be a random edge from x to an unexplored vertex y, chosen with probability inversely proportional to its edge weight ;
      $y \leftarrow$ explored ;
      push y to S ;
      $B(y) \leftarrow B(y) + 1$;
      inner-loop $\leftarrow$ inner-loop $+1$;
      $x \leftarrow y$;
    while (S is nonempty) **do**
      pop $x \leftarrow S$;
      $x \leftarrow$ unexplored ;
    outer-loop $\leftarrow$ outer-loop $+1$;
  foreach (vertex $v \in V$) **do**
    Output approximate k-path centrality of $v \leftarrow B(v) \cdot \frac{kn}{\ell}$;
end

Algorithm 9: Randomized Approximation k-Path

Analysis: We give the analysis for the case of unweighted graphs; the analysis for weighted graphs is similar. Let v be a vertex in the unweighted graph $G(V, E)$. Define, for $1 \leq i \leq \ell$,

$$X_i' = \begin{cases} 1 & \text{if the } i^{th} \text{ random walk passes through } v, \\ 0 & \text{otherwise.} \end{cases}$$

It is easy to see that when the algorithm terminates $B(v) = \sum_{i=1}^{\ell} X_i'$.

$$\begin{aligned} E[X_i'] &= \Pr(X_i' = 1) \\ &= \frac{1}{kn} \sum_{s \in V} \sum_{1 \le \ell \le k} \Pr(\text{a random walk of length } \ell \text{ starts from } s \text{ and passes through } v) \\ &= \frac{1}{kn} \cdot 0 + \frac{1}{kn} \sum_{s \neq v} \sum_{1 \le \ell \le k} \frac{p_v(s;\ell)}{p(s;\ell)} \\ &= \frac{1}{kn} \mathcal{C}_k(v). \end{aligned}$$

Let us define $X_i = knX_i'$. Notice that each X_i belongs to the range $[0, kn]$. Also, note that X_is are independent random variables.

Theorem 3.3 (Hoeffding's Bound) *If $X_1, X_2, ..., X_h$ are independent random variables with $a_i \le X_i \le b_i$ and $\mu = E\left[\frac{\sum_{i=1}^{h} X_i}{h}\right]$ is the expected mean, then, for every $\xi > 0$, we have*

$$\Pr\left(\left|\frac{\sum_{i=1}^{h} X_i}{h} - \mu\right| \ge \xi\right) \le 2e^{-2h^2\xi^2 / \sum_{i=1}^{h}(b_i - a_i)^2}.$$

The output of the algorithm is $\sum_{i=1}^{\ell} X_i/\ell$. Applying Hoeffding bound, we get

$$\Pr\left(\left|\frac{\sum_{i=1}^{\ell} X_i}{\ell} - \mathcal{C}_k(v)\right| \ge \xi\right) \le 2e^{-2\ell^2\xi^2/(\ell k^2 n^2)} = 2e^{-2\ell\xi^2/(k^2n^2)}.$$

Suppose that $\xi = n^{\frac{1}{2}+\alpha}$. Then, the probability of error is at most $2e^{-2\ell n^{1+2\alpha}/(k^2n^2)} = 2e^{-2\ell/(k^2n^{1-2\alpha})}$. This probability of error can be made at most $1/n^3$ by making $e^{-2\ell/(k^2n^{1-2\alpha})} \le e^{-\ln(2n^3)}$, or $\ell \ge k^2 n^{1-2\alpha}(\frac{3}{2}\ln n + \frac{1}{2}\ln 2)$. Therefore, for our algorithm we can set $\ell = 2k^2 n^{1-2\alpha} \ln n$, where the additive error ξ is $n^{\frac{1}{2}+\alpha}$.

In general, if the additive error is ξ, then we can use $\ell \ge (2k^2n^2 \ln n)/\xi^2$, which results in the running time $O((k^3n^2 \ln n)/\xi^2)$ of the randomized algorithm. Thus, for $\xi = n^{1/2+\alpha}$, the randomized algorithm takes $O(k^3 n^{1-2\alpha} \ln n)$ time. ∎

CHAPTER 4

RESULTS

In this chapter we compare Brandes' betweenness centrality algorithm [Bra01] with our approximation algorithm for k-path centrality. We conducted an extensive set of experiments on both synthetic (i.e., randomly generated) and real networks. For all of our experiments, we use values $\alpha = 0.01$ and $k = \ln(n+m)$, where n is the number of vertices and m is the number of edges in the graph. All the experiments in this thesis were done on the Research Computing cluster at the University of South Florida. Research Computing cluster consists of 316 nodes with a total of 2064 cores and the total memory is 4.812 Terabytes. We refer the reader to the Research Computing web page [Res10] for further details about the research environment used in this thesis.

4.1 Randomly Generated Networks

Our first experiment compares the running time of Brandes' betweenness centrality algorithm with our approximation algorithm for k-path centrality. For this experiment, we created 4500 undirected, unweighted random graphs with varying number of vertices and varying density. The density d of a graph is defined as the ratio of the number of edges in the graph and the total possible number of edges, i.e., $d = m/\binom{n}{2}$. We used a simple method to construct random graphs with a given number of vertices n and density d. First we compute the corresponding number of edges m, given the number of vertices n and density d, using the relation $m = d \cdot \binom{n}{2}$. Then, to generate a random graph $G(V, E)$ that has n vertices and m edges, we add m random edges to the initial empty set of edges as follows: uniformly at random pick two vertices from V and connect them by an edge.

Figure 4.1 shows a comparison of the running times (in seconds) between Brandes' betweenness centrality algorithm and our randomized approximation algorithm for k-path centrality

on undirected, unweighted random graphs. The random graphs are generated using the randomized sampling procedure described above. In this plot, the number of vertices vary from 100 to 5000 in increments of 100, and density varies from 10% to 90% in increments of 10.

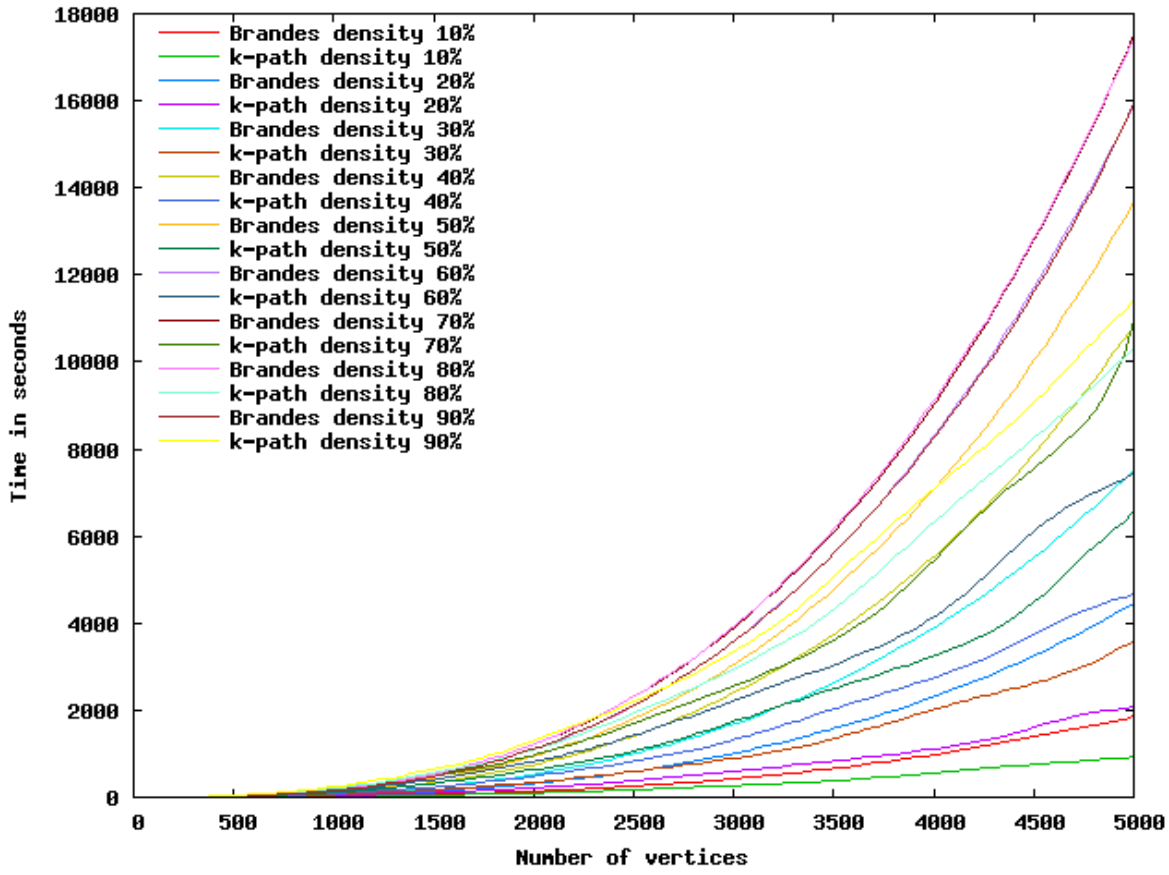

Figure 4.1 Plot showing running times (in seconds) of Brandes' algorithm and our randomized approximation algorithm

We can clearly see that the running time of Brandes' betweenness centrality algorithm is much higher than the running time of our approximation algorithm for k-path centrality when both algorithms are tested on the same random graph. This demonstrates that our algorithm yields a significant speedup in running time over Brandes' betweenness centrality algorithm. Furthermore, this holds true for any graph, directed or undirected and weighted or unweighted.

In our second set of experiments, we compare the values of exact betweenness centrality with the corresponding values of approximate k-path centrality in random graphs. It is inter-

esting to see that in all our plots for random graphs with density at most 60%, there is a nice linear correlation between the exact betweenness values and the corresponding approximate k-path centrality values. We now present these results for random networks using Tables 4.1 through 4.6 and Figures 4.2 through 4.7. Each table will show the information about a network, the computation times of algorithms, and the correlation between betweenness and approximate k-path centrality values. Each figure (corresponding to a table as well as an experiment) will show the scatter plots of exact betweenness centrality values with respect to the approximate k-path centrality values of all vertices of the network. Since experimentation on networks with 5000 vertices was feasible within our computational requirements, we present below some results for random networks in which the number of vertices is 5000 and the density varies from 10% to 60% in increments of 10.

Table 4.1 Summary information of a random network with 5000 vertices and 10% density

Number of vertices (n)	$5,000$
Number of edges (m)	$1,249,750$
Density ($d = m/\binom{n}{2}$)	10%
Directed or undirected	undirected
Weighted or unweighted	unweighted
Running time of Brandes' betweenness centrality algorithm	$1,334$ seconds
Alpha	0.01
Path length ($\ln(m+n)$)	14
Running time of our k-path centrality algorithm	488 seconds
Correlation coefficient between exact betweenness values and approximate k-path centrality values over all vertices	0.9868
Percentage of vertices common to both the top 10 betweenness values and the top 10 k-path centrality values	100%
Percentage of vertices common to both the top 20 betweenness values and the top 20 k-path centrality values	80%
Percentage of vertices common to both the top 50 betweenness values and the top 50 k-path centrality values	80%
Percentage of vertices common to both the top 100 betweenness values and the top 100 k-path centrality values	85%

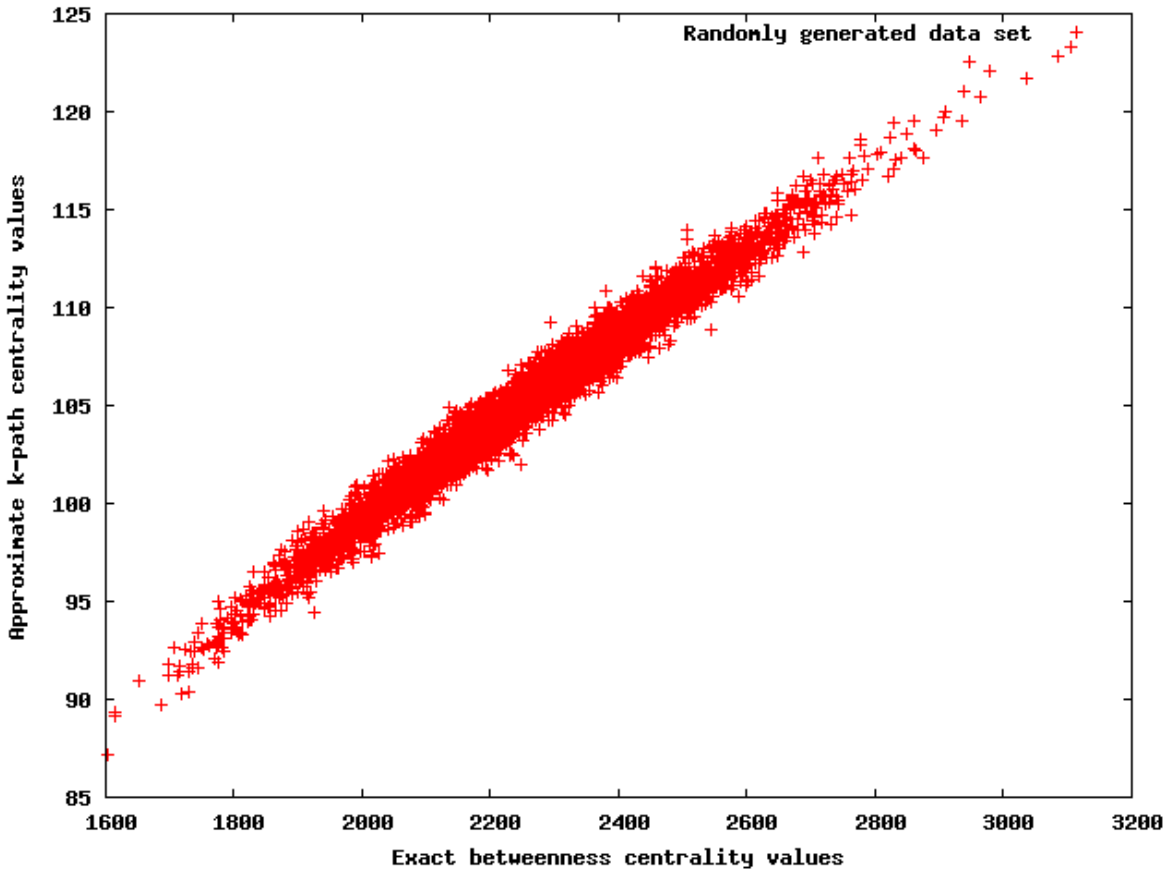

Figure 4.2 Scatter plot for a random network with 5000 vertices and 10% density

Table 4.2 Summary information of a random network with 5000 vertices and 20% density

Number of vertices (n)	$5,000$
Number of edges (m)	$2,499,500$
Density ($d = m/\binom{n}{2}$)	20%
Directed or undirected	undirected
Weighted or unweighted	unweighted
Running time of Brandes' betweenness centrality algorithm	$3,218$ seconds
Alpha	0.01
Path length ($\ln(m+n)$)	15
Running time of our k-path centrality algorithm	$1,285$ seconds
Correlation coefficient between exact betweenness values and approximate k-path centrality values over all vertices	0.9786
Percentage of vertices common to both the top 10 betweenness values and the top 10 k-path centrality values	80%
Percentage of vertices common to both the top 20 betweenness values and the top 20 k-path centrality values	70%
Percentage of vertices common to both the top 50 betweenness values and the top 50 k-path centrality values	74%
Percentage of vertices common to both the top 100 betweenness values and the top 100 k-path centrality values	79%

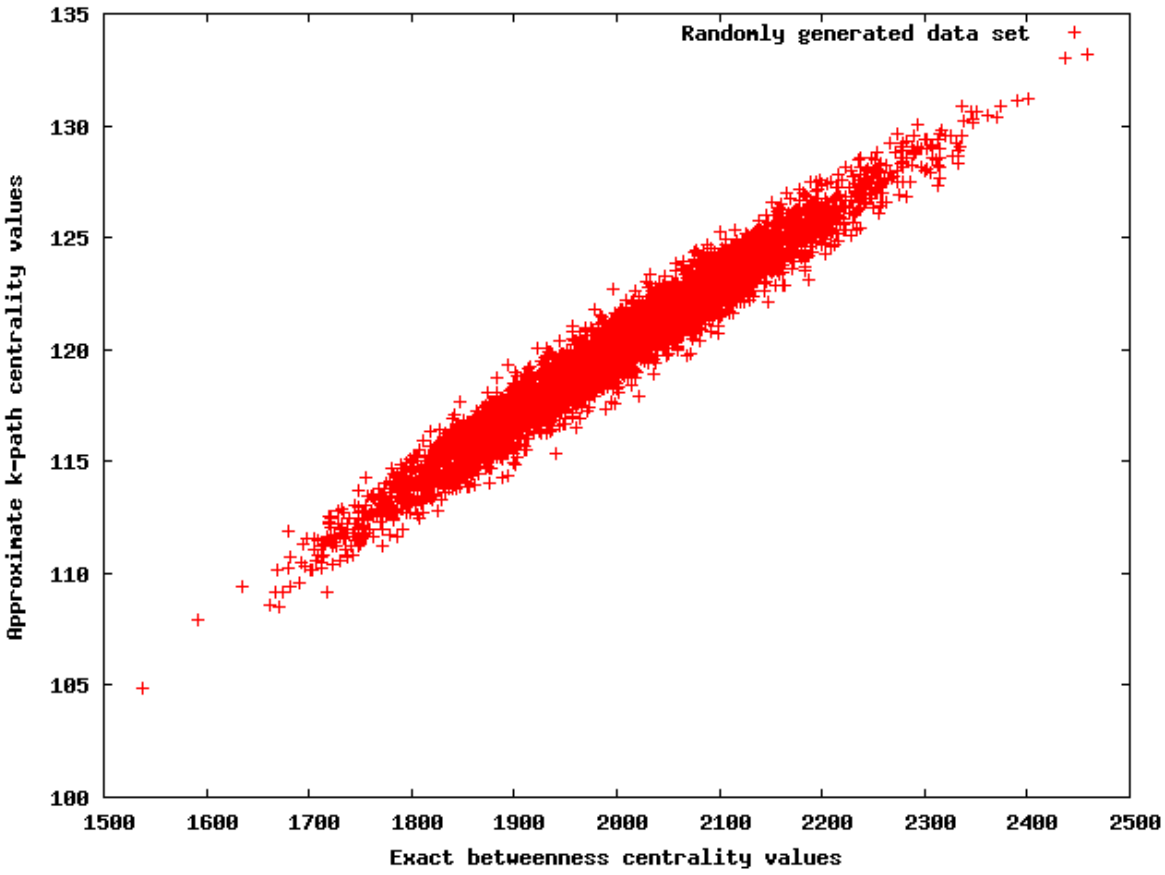

Figure 4.3 Scatter plot for a random network with 5000 vertices and 20% density

Table 4.3 Summary information of a random network with 5000 vertices and 30% density

Number of vertices (n)	$5,000$
Number of edges (m)	$3,749,250$
Density ($d = m/\binom{n}{2}$)	30%
Directed or undirected	undirected
Weighted or unweighted	unweighted
Running time of Brandes' betweenness centrality algorithm	$5,254$ seconds
Alpha	0.01
Path length ($\ln(m+n)$)	15
Running time of our k-path centrality algorithm	$1,758$ seconds
Correlation coefficient between exact betweenness values and approximate k-path centrality values over all vertices	0.9683
Percentage of vertices common to both the top 10 betweenness values and the top 10 k-path centrality values	60%
Percentage of vertices common to both the top 20 betweenness values and the top 20 k-path centrality values	70%
Percentage of vertices common to both the top 50 betweenness values and the top 50 k-path centrality values	70%
Percentage of vertices common to both the top 100 betweenness values and the top 100 k-path centrality values	75%

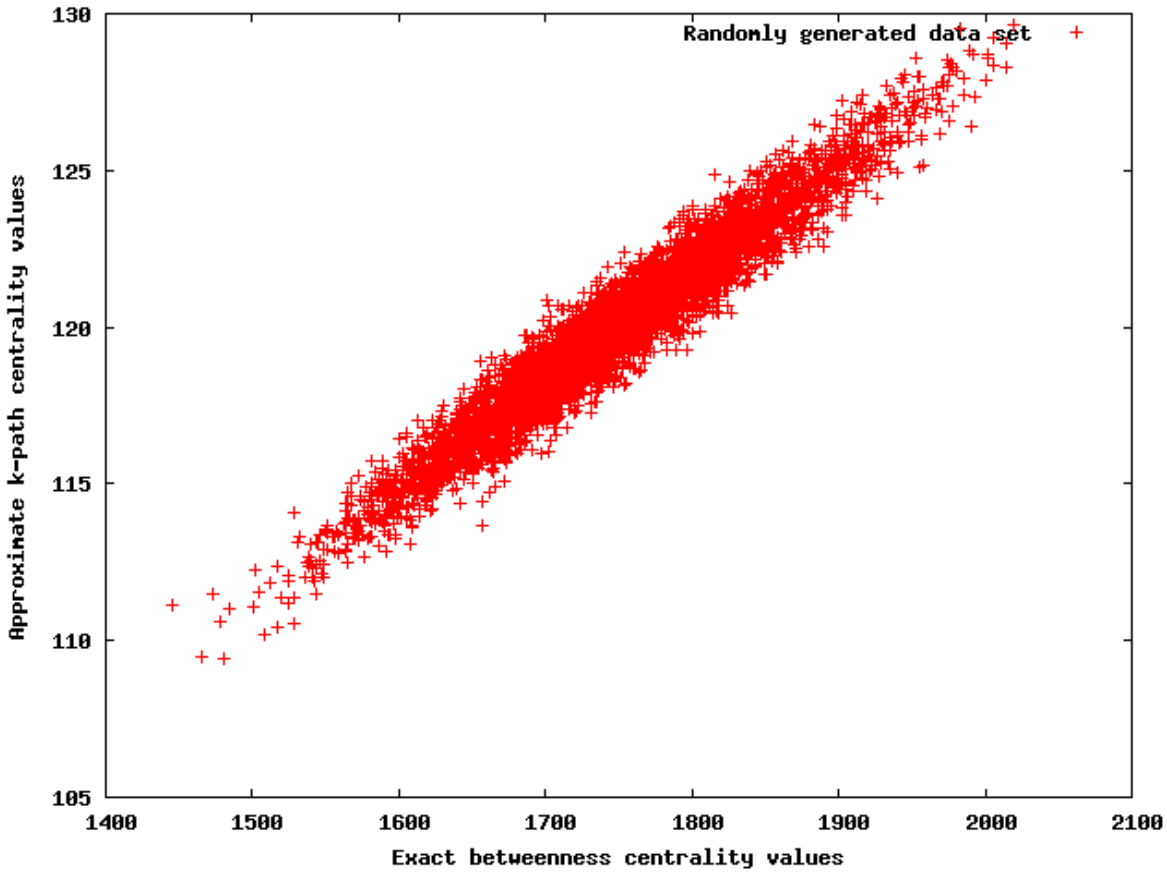

Figure 4.4 Scatter plot for a random network with 5000 vertices and 30% density

Table 4.4 Summary information of a random network with 5000 vertices and 40% density

Number of vertices (n)	$5,000$
Number of edges (m)	$4,999,000$
Density ($d = m/\binom{n}{2}$)	40%
Directed or undirected	undirected
Weighted or unweighted	unweighted
Running time of Brandes' betweenness centrality algorithm	$7,535$ seconds
Alpha	0.01
Path length ($\ln(m+n)$)	15
Running time of our k-path centrality algorithm	$2,375$ seconds
Correlation coefficient between exact betweenness values and approximate k-path centrality values over all vertices	0.9509
Percentage of vertices common to both the top 10 betweenness values and the top 10 k-path centrality values	50%
Percentage of vertices common to both the top 20 betweenness values and the top 20 k-path centrality values	50%
Percentage of vertices common to both the top 50 betweenness values and the top 50 k-path centrality values	62%
Percentage of vertices common to both the top 100 betweenness values and the top 100 k-path centrality values	71%

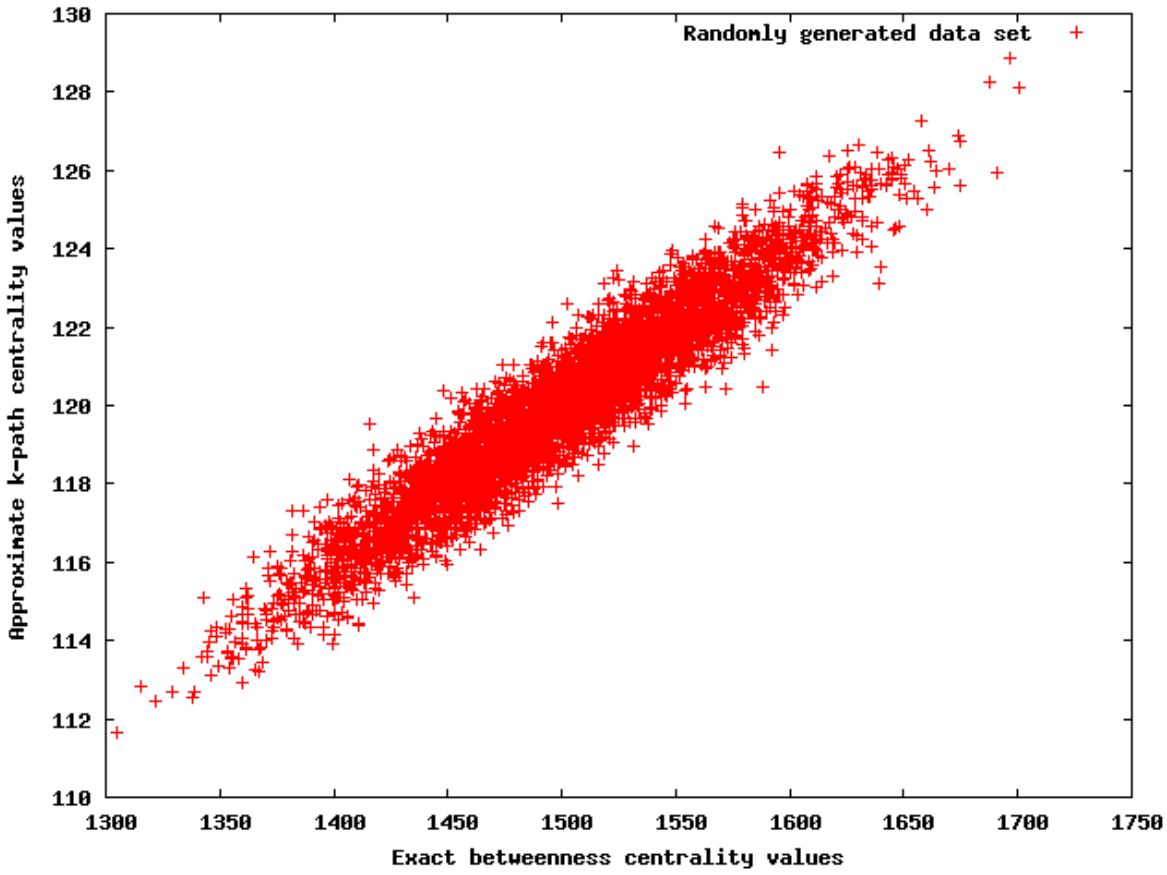

Figure 4.5 Scatter plot for a random network with 5000 vertices and 40% density

Table 4.5 Summary information of a random network with 5000 vertices and 50% density

Number of vertices (n)	$5,000$
Number of edges (m)	$6,248,750$
Density ($d = m/\binom{n}{2}$)	50%
Directed or undirected	undirected
Weighted or unweighted	unweighted
Running time of Brandes' betweenness centrality algorithm	$9,749$ seconds
Alpha	0.01
Path length ($\ln(m+n)$)	16
Running time of our k-path centrality algorithm	$4,029$ seconds
Correlation coefficient between exact betweenness values and approximate k-path centrality values over all vertices	0.9451
Percentage of vertices common to both the top 10 betweenness values and the top 10 k-path centrality values	50%
Percentage of vertices common to both the top 20 betweenness values and the top 20 k-path centrality values	65%
Percentage of vertices common to both the top 50 betweenness values and the top 50 k-path centrality values	60%
Percentage of vertices common to both the top 100 betweenness values and the top 100 k-path centrality values	68%

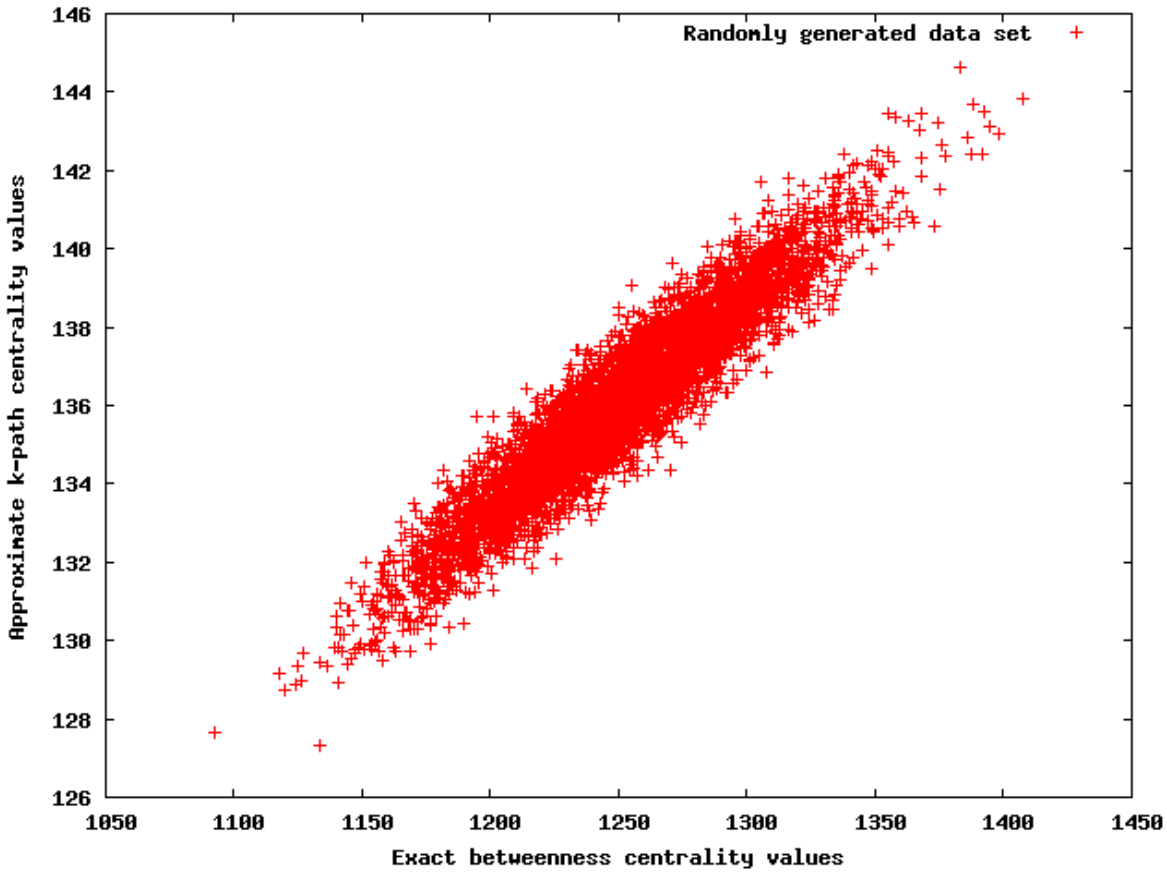

Figure 4.6 Scatter plot for a random network with 5000 vertices and 50% density

Table 4.6 Summary information of a random network with 5000 vertices and 60% density

Number of vertices (n)	$5,000$
Number of edges (m)	$7,498,500$
Density ($d = m/\binom{n}{2}$)	60%
Directed or undirected	undirected
Weighted or unweighted	unweighted
Running time of Brandes' betweenness centrality algorithm	$11,451$ seconds
Alpha	0.01
Path length ($\ln(m+n)$)	16
Running time of our k-path centrality algorithm	$4,624$ seconds
Correlation coefficient between exact betweenness values and approximate k-path centrality values over all vertices	0.9290
Percentage of vertices common to both the top 10 betweenness values and the top 10 k-path centrality values	60%
Percentage of vertices common to both the top 20 betweenness values and the top 20 k-path centrality values	45%
Percentage of vertices common to both the top 50 betweenness values and the top 50 k-path centrality values	70%
Percentage of vertices common to both the top 100 betweenness values and the top 100 k-path centrality values	69%

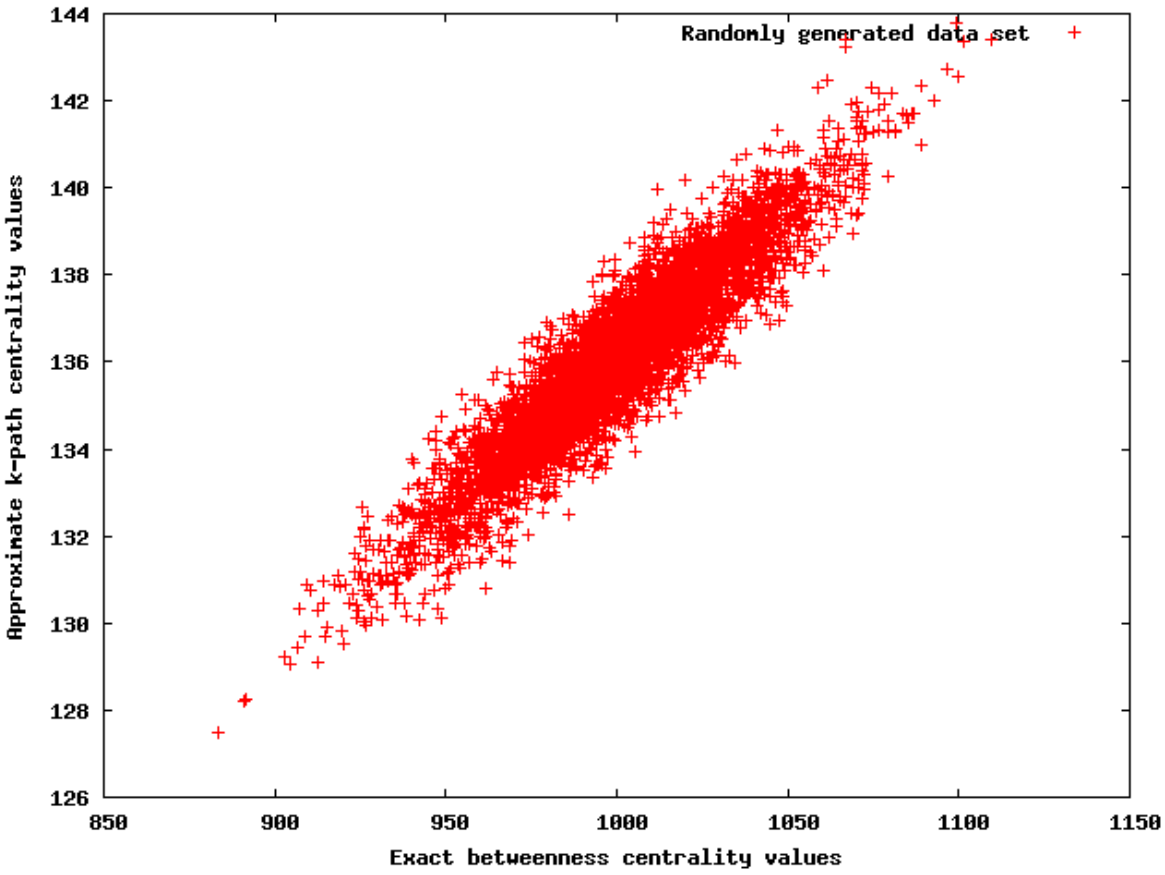

Figure 4.7 Scatter plot for a random network with 5000 vertices and 60% density

4.2 Real Networks

Next we experiment with several real network data obtained from various resources [DH97, BM06, BJMO07, New08]. We highly acknowledge the effort of those individuals who obtained these data sets from different sources and making them available to the public. The information about various data sets is stated below.

- The Zachary's Karate Club data set [Zac77, New08]: According to Wayne Zachary in 1977, it contains a network of friendships between members of a karate club at a US university. Vertices represent members at the karate club and edges represent relationships between members. This is an undirected, unweighted graph with 34 vertices and 78 edges that has been cited in some literature [New04, LN05, DFLJ07, GL08, MG09, MMO09, CHL10]. The summary information of this data set is shown in Table 4.7. The scatter plot showing approximate k-path centrality values in relation to the corresponding betweenness values for this data set is shown in Figure 4.8.

- The "Les Miserables" data set [Knu93, New08]: It contains data of a weighted, undirected network of co-appearances of characters in Victor Hugo's novel "Les Miserables". Vertices represent characters and edges connect any pair of characters that appear in the same chapter of the book. The number of such co-appearances indicate the edge weights. There are 77 vertices and 254 edges in this network. This data set has been cited in some literature related to community structures [HJ08, MMO09, CHL10]. The summary information of this data set is shown in Table 4.8. The scatter plot showing approximate k-path centrality values in relation to the corresponding betweenness values for this data set is shown in Figure 4.9.

- The Word Adjacencies data set [New06, New08]: It contains data of a network of 112 vertices that represent commonly occurring adjectives and nouns in the novel "David Copperfield" by Charles Dickens, as described by Mark Newman. The edges connect any pair of words that appear adjacent to each other at any point in the text. The network is unweighted, undirected and consists of 425 edges. This data set has been cited in some papers by Newman and others [New06, NL07, BRTC08]. The summary

information of this data set is shown in Table 4.9. The scatter plot showing approximate k-path centrality values in relation to the corresponding betweenness values for this data set is shown in Figure 4.10.

- Condensed Matter data sets [New01b, New08]: There are three Condensed Matter Collaboration data sets used in this thesis. They consist of weighted, undirected networks of coauthorships between scientists posting preprints on the Condensed Matter E-Print Archive starting from January 1, 1995. The weight of an edge between two scientists represents the number of coauthorships between them. The end dates of collection of these three data sets are December 31, 1999, June 30, 2003 and March 31, 2005. These networks have been cited in some literature [New01a, New01b, New01c, BRTC08, JGH10]. The summary information for these three data sets are shown in Tables 4.10, 4.11, and 4.12. The scatter plots showing approximate k-path centrality values in relation to the corresponding betweenness values for these data sets are shown in Figures 4.11, 4.12, and 4.13.

- High-Energy Theory data set [New01b, New08]: According to Mark Newman, this data set consists of a weighted, undirected network of coauthorships between 8361 scientists posting preprints on the High-Energy Theory E-Print Archive between January 1, 1995 and December 31, 1999. This network consists of 8361 vertices and 15751 edges. The weight of an edge between two scientists represents the number of coauthorships between them. This network has been cited in some literature [New01a, New01b, BRTC08]. The summary information of this data set is shown in Table 4.13. The scatter plot showing approximate k-path centrality values in relation to the corresponding betweenness values for this data set is shown in Figure 4.14.

- Internet data set [New08]: This data set represents an unweighted, undirected network with 22963 vertices and 48436 edges. Using BGP tables posted by the University of Oregon Route Views Project, Mark Newman created this data set July 22, 2006. It consists of a symmetrized snapshot of the structure of the Internet at the level of autonomous systems. This network has been cited in some literature [BRTC08, GCZ09]. The summary information of this data set is shown in Table 4.14. The scatter plot show-

ing approximate k-path centrality values in relation to the corresponding betweenness values for this data set is shown in Figure 4.15.

- Yeast data set [BCC+03, BM06]: This data set represents protein (yeasts) as vertices and protein interactions as edges. There are 1870 type of proteins and 8960 number of interactions among two proteins. This is an unweighted, undirected network, which has been cited in some papers [BCC+03, BRTC08]. The summary information of this data set is shown in Table 4.15. The scatter plot showing approximate k-path centrality values in relation to the corresponding betweenness values for this data set is shown in Figure 4.16.

- Kazaa data sets [IRF04]: Kazaa is a popular peer-to-peer file-sharing system. As of June 2003, there are more than 4 million estimated concurrent users. Here, information about the files requested for download is used to build a graph using the users as vertices of the graph. Two users are connected in this graph if they have the same download during some interval. Three data sets are formed in three different time intervals and we refer the reader to the paper [IRF04] for further details. The summary information of these data sets are shown in Tables 4.16, 4.17, and 4.18. The scatter plot showing approximate k-path centrality values in relation to the corresponding betweenness values for these data sets are shown in Figures 4.17, 4.18, and 4.19.

- Computational Geometry data set [BM06]: This data set consists of author collaboration in computational geometry, where vertices represent authors. If two authors wrote a common work (book, paper, etc.), then we have an edge between those two vertices and the edge weight is the number of such common works. This undirected, weighted graph has 6158 vertices and 11898 edges. We refer the reader to the Pajek web page [BM06] for further details. The summary information of this data set is shown in Table 4.19. The scatter plot showing approximate k-path centrality values in relation to the corresponding betweenness values for this data set is shown in Figure 4.20.

- Pajek data sets [BM06]: The last five data sets consist of citation networks where vertices represent people and edges represent relationships between two people. The

number of vertices varies from 233 to 6651 and the number of edges varies from 994 to 54253. All five graphs are undirected and unweighted, and we refer the reader to the Pajek web page [BM06] for further details. These data sets have been cited in some literature [Bat03, PPZ10]. The summary information of these data sets are shown in Tables 4.20 through 4.24. The scatter plots showing approximate k-path centrality values in relation to the corresponding betweenness values for these data sets are shown in Figures 4.21 through 4.25.

Table 4.7 Summary information of Zachary's Karate Club data set

Number of vertices (n)	34
Number of edges (m)	78
Density ($d = m/\binom{n}{2}$)	13.904%
Directed or undirected	undirected
Weighted or unweighted	unweighted
Running time of Brandes' betweenness centrality algorithm	< 1 seconds
Alpha	0.01
Path length ($\ln(m+n)$)	5
Running time of our k-path centrality algorithm	< 1 seconds
Correlation coefficient between exact betweenness values and approximate k-path centrality values over all vertices	0.9129
Percentage of vertices common to both the top 5 betweenness values and the top 5 k-path centrality values	80%
Percentage of vertices common to both the top 10 betweenness values and the top 10 k-path centrality values	70%
Percentage of vertices common to both the top 20 betweenness values and the top 20 k-path centrality values	85%

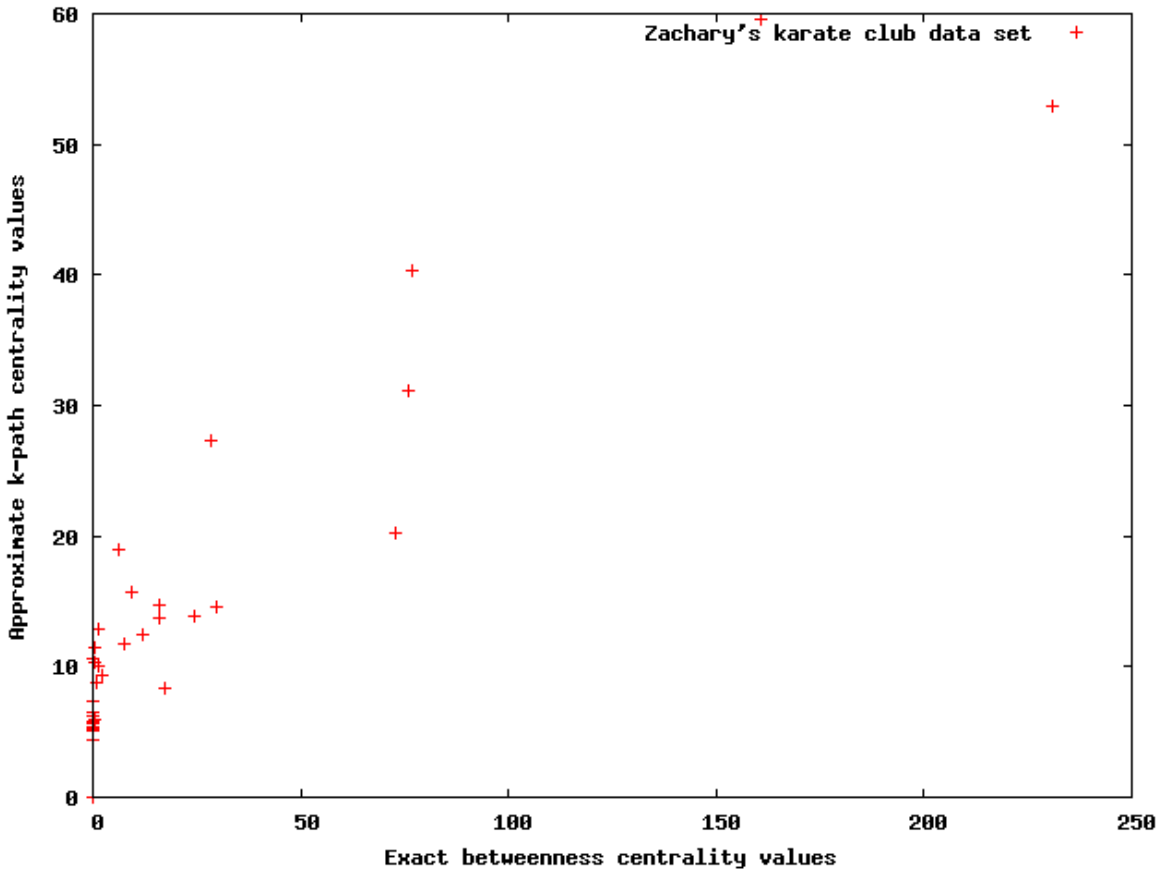

Figure 4.8 Scatter plot for Zachary's Karate Club data set

Table 4.8 Summary information of Les Miserables data set

Number of vertices (n)	77
Number of edges (m)	254
Density ($d = m/\binom{n}{2}$)	8.681%
Directed or undirected	undirected
Weighted or unweighted	weighted
Running time of Brandes' betweenness centrality algorithm	$<$ 1 seconds
Alpha	0.01
Path length ($\ln(m+n)$)	6
Running time of our k-path centrality algorithm	$<$ 1 seconds
Correlation coefficient between exact betweenness values and approximate k-path centrality values over all vertices	0.9061
Percentage of vertices common to both the top 5 betweenness values and the top 5 k-path centrality values	80%
Percentage of vertices common to both the top 10 betweenness values and the top 10 k-path centrality values	80%
Percentage of vertices common to both the top 20 betweenness values and the top 20 k-path centrality values	80%

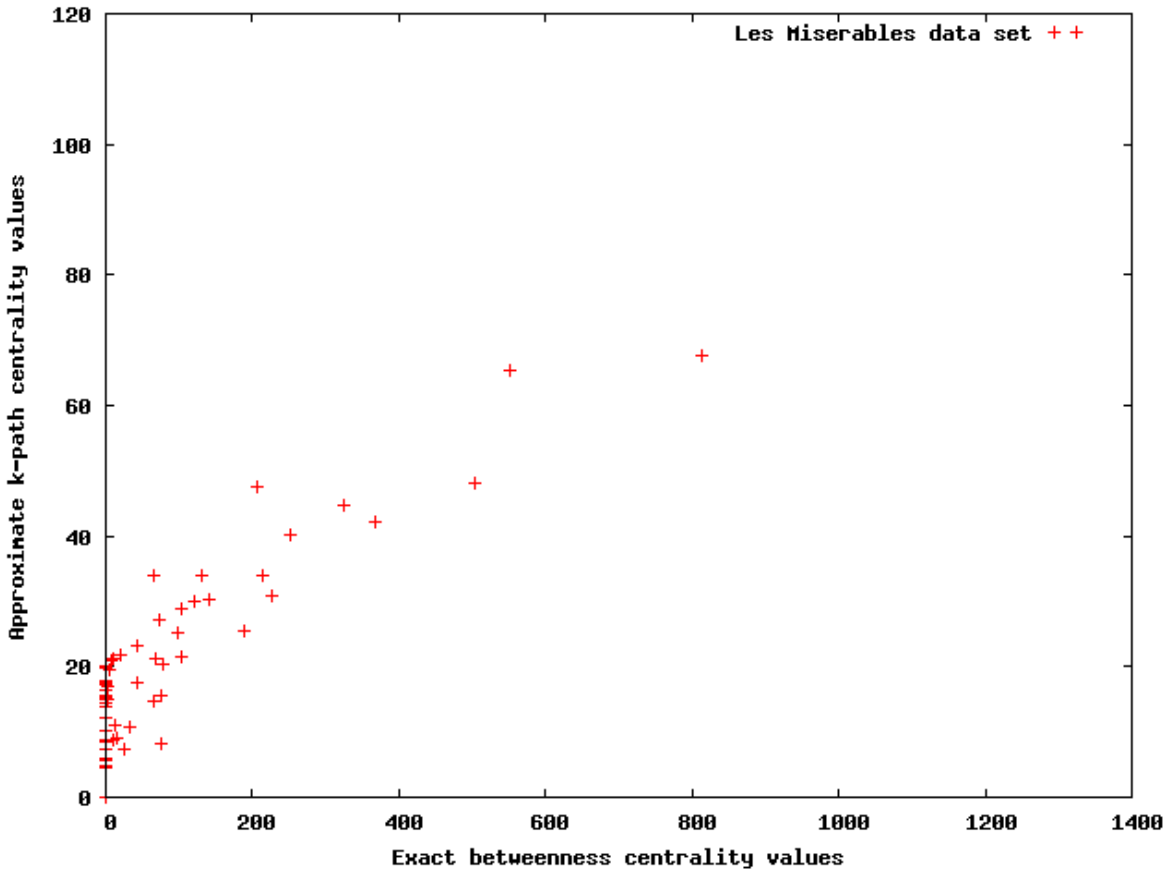

Figure 4.9 Scatter plot for Les Miserables data set

Table 4.9 Summary information of Word Adjacencies data set

Number of vertices (n)	112
Number of edges (m)	425
Density ($d = m/\binom{n}{2}$)	6.837%
Directed or undirected	undirected
Weighted or unweighted	unweighted
Running time of Brandes' betweenness centrality algorithm	< 1 seconds
Alpha	0.01
Path length ($\ln(m+n)$)	6
Running time of our k-path centrality algorithm	< 1 seconds
Correlation coefficient between exact betweenness values and approximate k-path centrality values over all vertices	0.9268
Percentage of vertices common to both the top 5 betweenness values and the top 5 k-path centrality values	80%
Percentage of vertices common to both the top 10 betweenness values and the top 10 k-path centrality values	90%
Percentage of vertices common to both the top 20 betweenness values and the top 20 k-path centrality values	70%

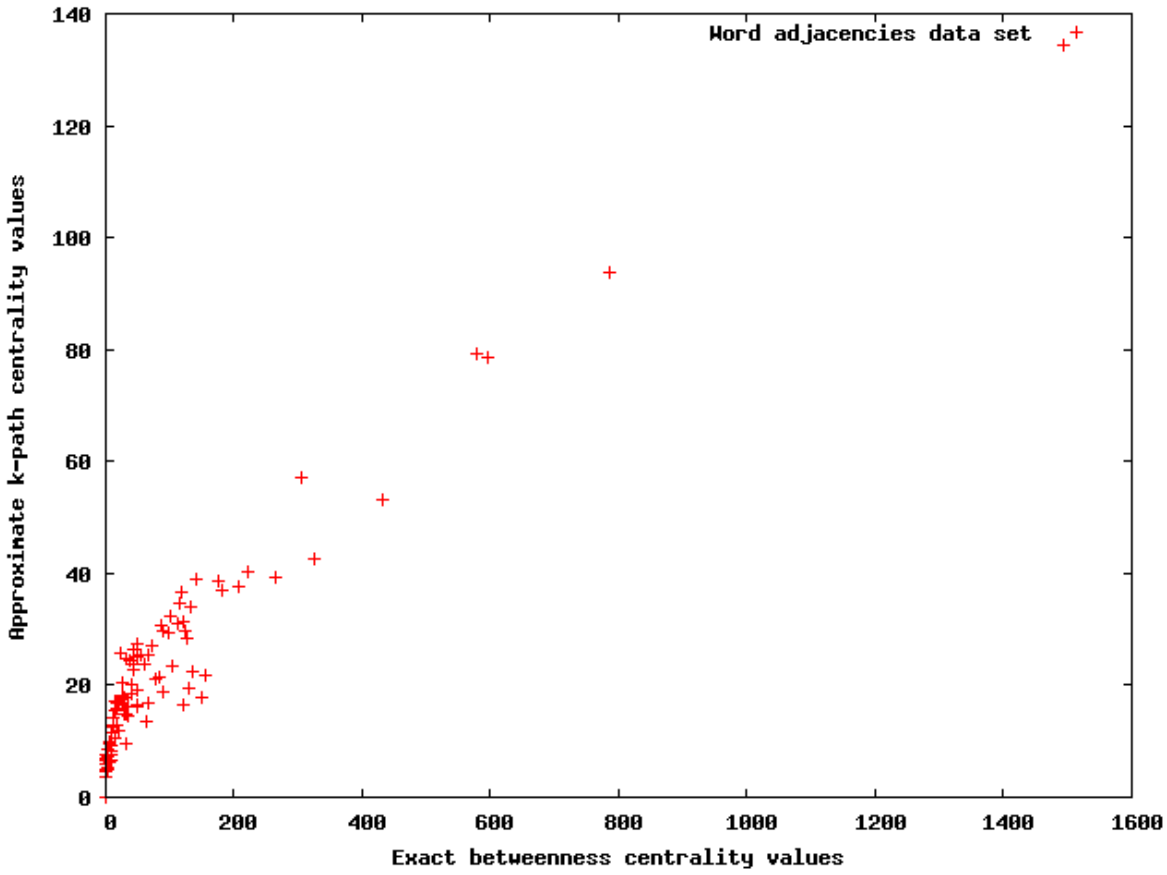

Figure 4.10 Scatter plot for Word Adjacencies data set

Table 4.10 Summary information of Condensed Matter 1999 data set

Number of vertices (n)	$16,726$
Number of edges (m)	$47,594$
Density ($d = m/\binom{n}{2}$)	0.034%
Directed or undirected	undirected
Weighted or unweighted	weighted
Running time of Brandes' betweenness centrality algorithm	640 seconds
Alpha	0.01
Path length ($\ln(m+n)$)	11
Running time of our k-path centrality algorithm	52 seconds
Correlation coefficient between exact betweenness values and approximate k-path centrality values over all vertices	0.6502
Percentage of vertices common to both the top 5 betweenness values and the top 5 k-path centrality values	60%
Percentage of vertices common to both the top 10 betweenness values and the top 10 k-path centrality values	60%
Percentage of vertices common to both the top 20 betweenness values and the top 20 k-path centrality values	45%
Percentage of vertices common to both the top 50 betweenness values and the top 50 k-path centrality values	56%
Percentage of vertices common to both the top 100 betweenness values and the top 100 k-path centrality values	56%

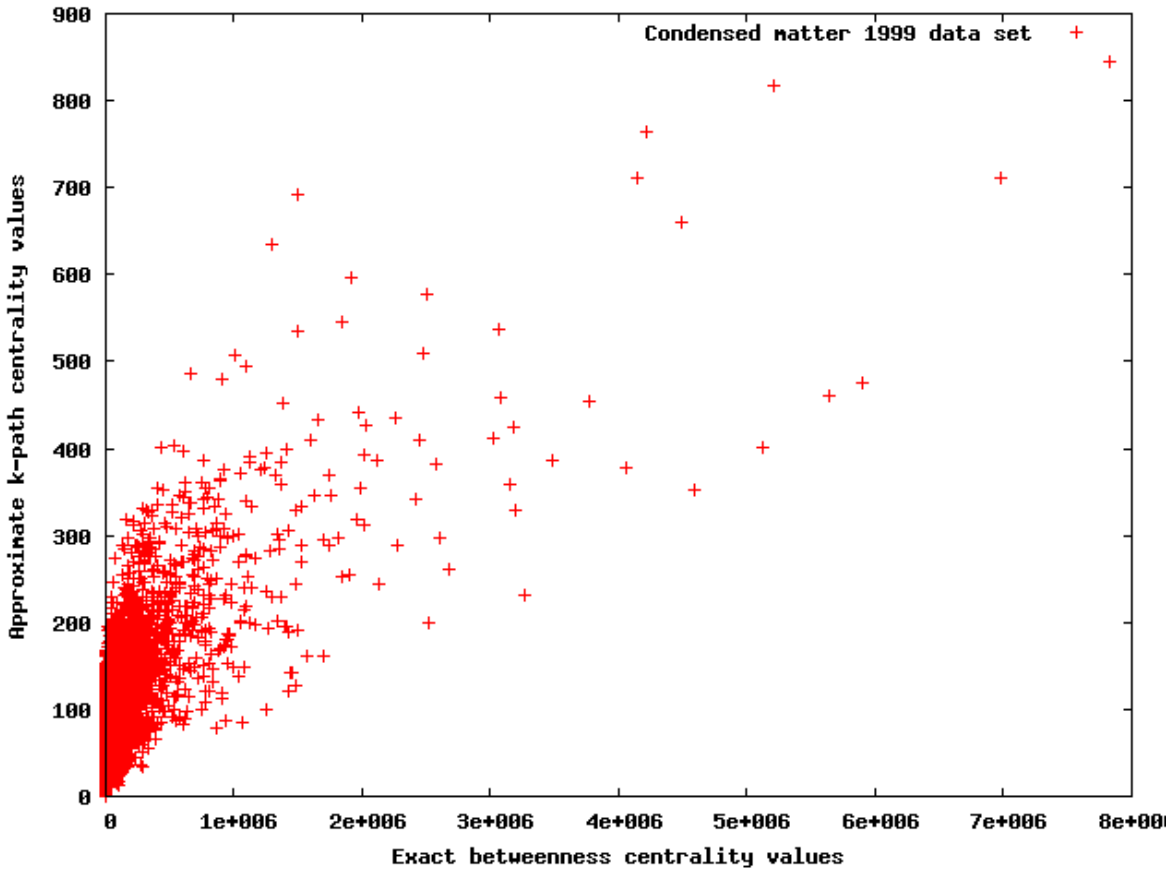

Figure 4.11 Scatter plot for Condensed Matter 1999 data set

Table 4.11 Summary information of Condensed Matter 2003 data set

Number of vertices (n)	$31,163$
Number of edges (m)	$120,029$
Density ($d = m/\binom{n}{2}$)	0.025%
Directed or undirected	undirected
Weighted or unweighted	weighted
Running time of Brandes' betweenness centrality algorithm	$2,884$ seconds
Alpha	0.01
Path length ($\ln(m+n)$)	12
Running time of our k-path centrality algorithm	183 seconds
Correlation coefficient between exact betweenness values and approximate k-path centrality values over all vertices	0.6920
Percentage of vertices common to both the top 5 betweenness values and the top 5 k-path centrality values	60%
Percentage of vertices common to both the top 10 betweenness values and the top 10 k-path centrality values	60%
Percentage of vertices common to both the top 20 betweenness values and the top 20 k-path centrality values	55%
Percentage of vertices common to both the top 50 betweenness values and the top 50 k-path centrality values	54%
Percentage of vertices common to both the top 100 betweenness values and the top 100 k-path centrality values	55%

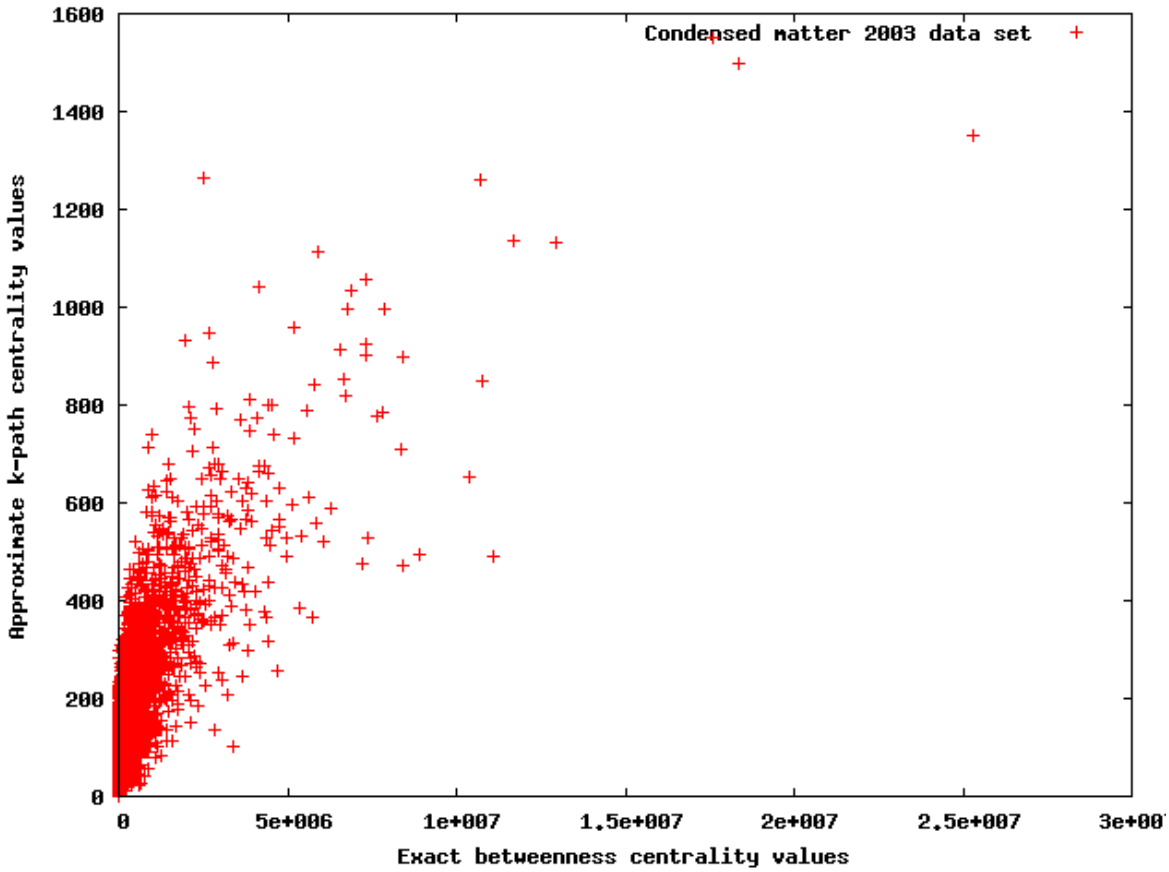

Figure 4.12 Scatter plot for Condensed Matter 2003 data set

Table 4.12 Summary information of Condensed Matter 2005 data set

Number of vertices (n)	$40,421$
Number of edges (m)	$175,693$
Density ($d = m/\binom{n}{2}$)	0.022%
Directed or undirected	undirected
Weighted or unweighted	weighted
Running time of Brandes' betweenness centrality algorithm	$5,405$ seconds
Alpha	0.01
Path length ($\ln(m+n)$)	12
Running time of our k-path centrality algorithm	311 seconds
Correlation coefficient between exact betweenness values and approximate k-path centrality values over all vertices	0.6982
Percentage of vertices common to both the top 5 betweenness values and the top 5 k-path centrality values	60%
Percentage of vertices common to both the top 10 betweenness values and the top 10 k-path centrality values	70%
Percentage of vertices common to both the top 20 betweenness values and the top 20 k-path centrality values	45%
Percentage of vertices common to both the top 50 betweenness values and the top 50 k-path centrality values	54%
Percentage of vertices common to both the top 100 betweenness values and the top 100 k-path centrality values	55%

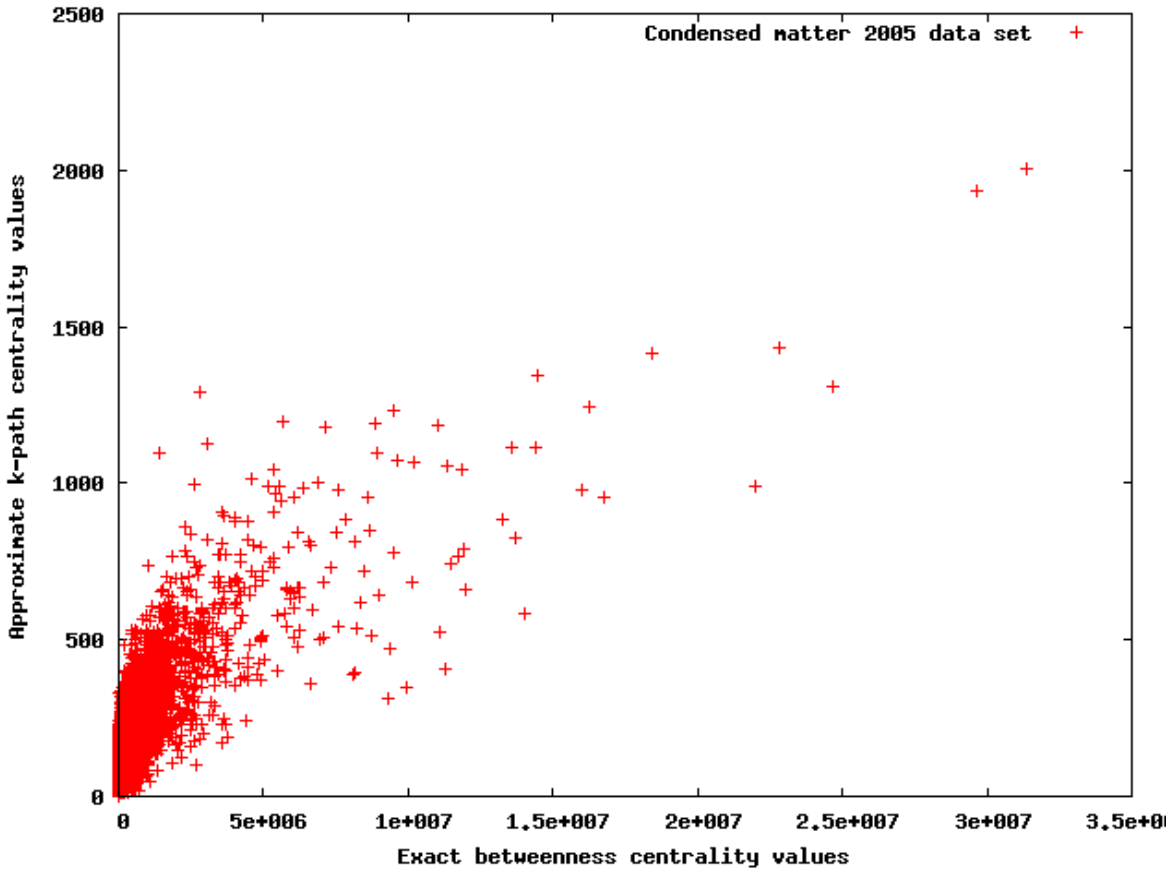

Figure 4.13 Scatter plot for Condensed Matter 2005 data set

Table 4.13 Summary information of High-Energy Theory data set

Number of vertices (n)	$8,361$
Number of edges (m)	$15,751$
Density ($d = m/\binom{n}{2}$)	0.045%
Directed or undirected	undirected
Weighted or unweighted	weighted
Running time of Brandes' betweenness centrality algorithm	109 seconds
Alpha	0.01
Path length ($\ln(m+n)$)	10
Running time of our k-path centrality algorithm	14 seconds
Correlation coefficient between exact betweenness values and approximate k-path centrality values over all vertices	0.7032
Percentage of vertices common to both the top 5 betweenness values and the top 5 k-path centrality values	40%
Percentage of vertices common to both the top 10 betweenness values and the top 10 k-path centrality values	60%
Percentage of vertices common to both the top 20 betweenness values and the top 20 k-path centrality values	50%
Percentage of vertices common to both the top 50 betweenness values and the top 50 k-path centrality values	42%
Percentage of vertices common to both the top 100 betweenness values and the top 100 k-path centrality values	54%

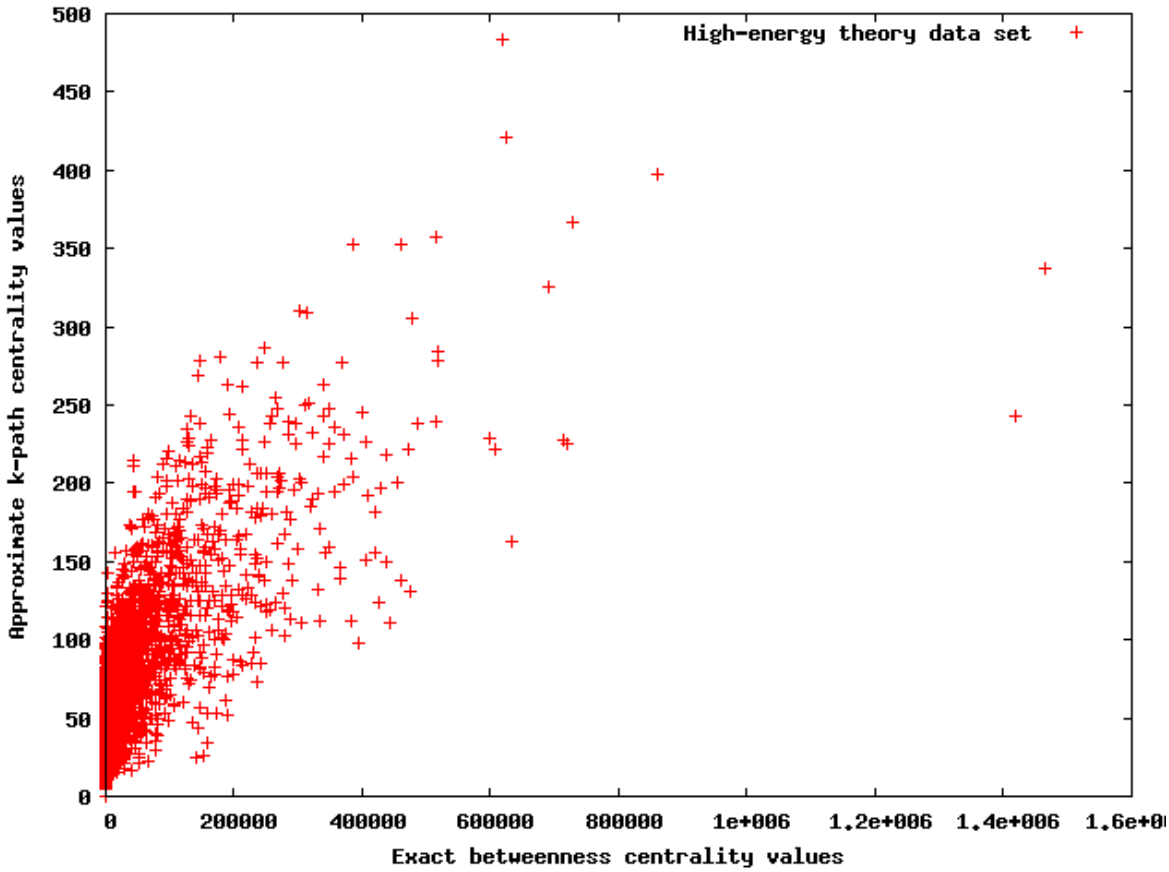

Figure 4.14 Scatter plot for High-Energy Theory data set

Table 4.14 Summary information of Internet data set

Number of vertices (n)	$22,963$
Number of edges (m)	$48,436$
Density ($d = m/\binom{n}{2}$)	0.018%
Directed or undirected	undirected
Weighted or unweighted	unweighted
Running time of Brandes' betweenness centrality algorithm	917 seconds
Alpha	0.01
Path length ($\ln(m+n)$)	11
Running time of our k-path centrality algorithm	462 seconds
Correlation coefficient between exact betweenness values and approximate k-path centrality values over all vertices	0.9563
Percentage of vertices common to both the top 5 betweenness values and the top 5 k-path centrality values	80%
Percentage of vertices common to both the top 10 betweenness values and the top 10 k-path centrality values	80%
Percentage of vertices common to both the top 20 betweenness values and the top 20 k-path centrality values	85%
Percentage of vertices common to both the top 50 betweenness values and the top 50 k-path centrality values	84%
Percentage of vertices common to both the top 100 betweenness values and the top 100 k-path centrality values	82%

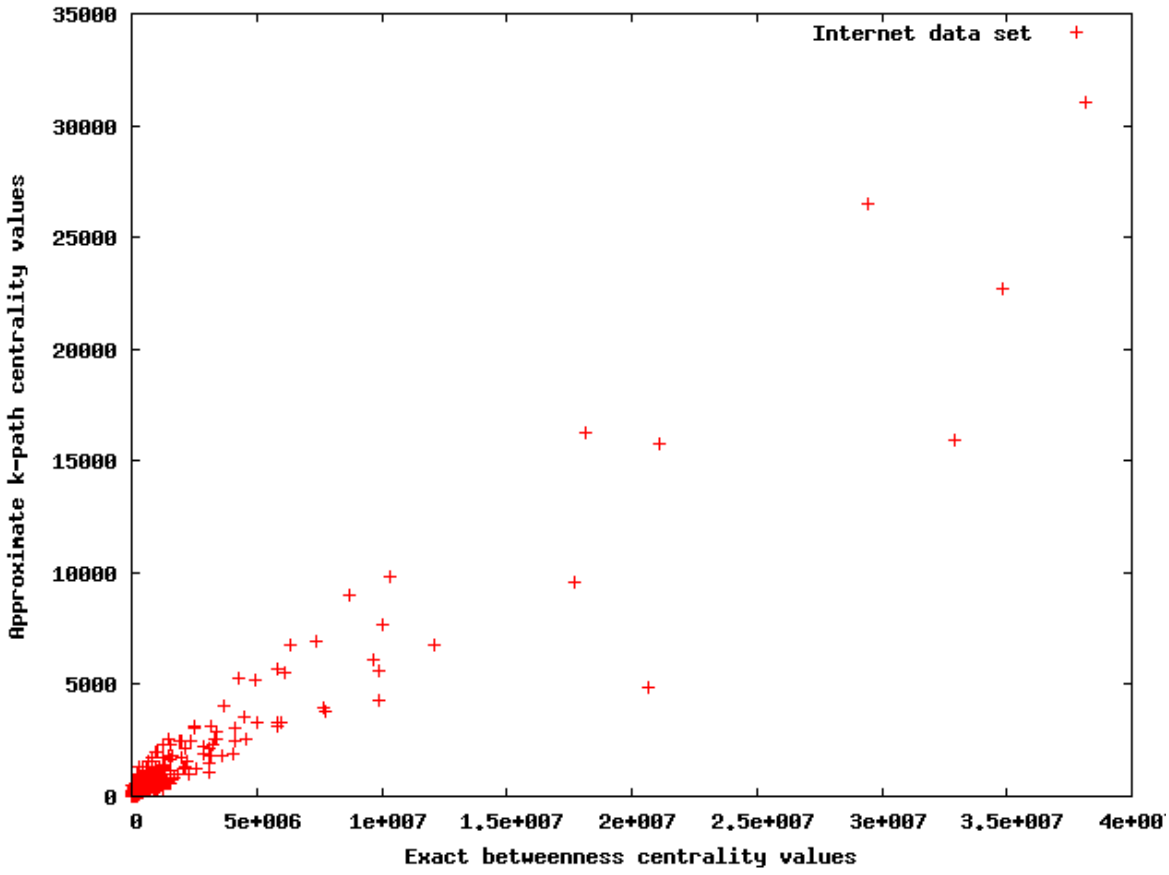

Figure 4.15 Scatter plot for Internet data set

Table 4.15 Summary information of Yeast data set

Number of vertices (n)	1,870
Number of edges (m)	8,960
Density ($d = m/\binom{n}{2}$)	0.513%
Directed or undirected	undirected
Weighted or unweighted	unweighted
Running time of Brandes' betweenness centrality algorithm	10 seconds
Alpha	0.01
Path length ($\ln(m+n)$)	9
Running time of our k-path centrality algorithm	2 seconds
Correlation coefficient between exact betweenness values and approximate k-path centrality values over all vertices	0.8600
Percentage of vertices common to both the top 5 betweenness values and the top 5 k-path centrality values	80%
Percentage of vertices common to both the top 10 betweenness values and the top 10 k-path centrality values	80%
Percentage of vertices common to both the top 20 betweenness values and the top 20 k-path centrality values	60%
Percentage of vertices common to both the top 50 betweenness values and the top 50 k-path centrality values	56%
Percentage of vertices common to both the top 100 betweenness values and the top 100 k-path centrality values	69%

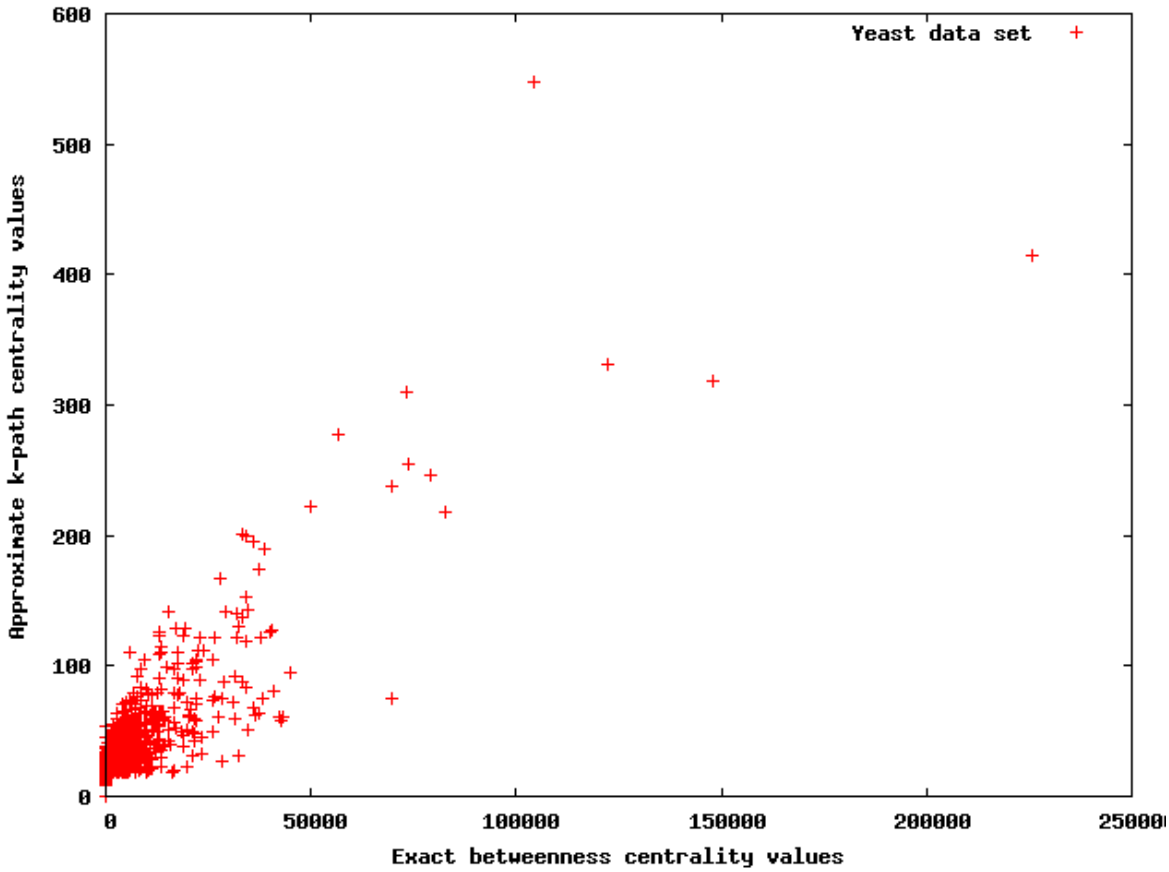

Figure 4.16 Scatter plot for Yeast data set

Table 4.16 Summary information of Kazaa data set 1

Number of vertices (n)	$1,400$
Number of edges (m)	$6,316$
Density ($d = m/\binom{n}{2}$)	0.645%
Directed or undirected	undirected
Weighted or unweighted	weighted
Running time of Brandes' betweenness centrality algorithm	6 seconds
Alpha	0.01
Path length ($\ln(m+n)$)	9
Running time of our k-path centrality algorithm	2 seconds
Correlation coefficient between exact betweenness values and approximate k-path centrality values over all vertices	0.8515
Percentage of vertices common to both the top 5 betweenness values and the top 5 k-path centrality values	80%
Percentage of vertices common to both the top 10 betweenness values and the top 10 k-path centrality values	80%
Percentage of vertices common to both the top 20 betweenness values and the top 20 k-path centrality values	70%
Percentage of vertices common to both the top 50 betweenness values and the top 50 k-path centrality values	80%
Percentage of vertices common to both the top 100 betweenness values and the top 100 k-path centrality values	80%

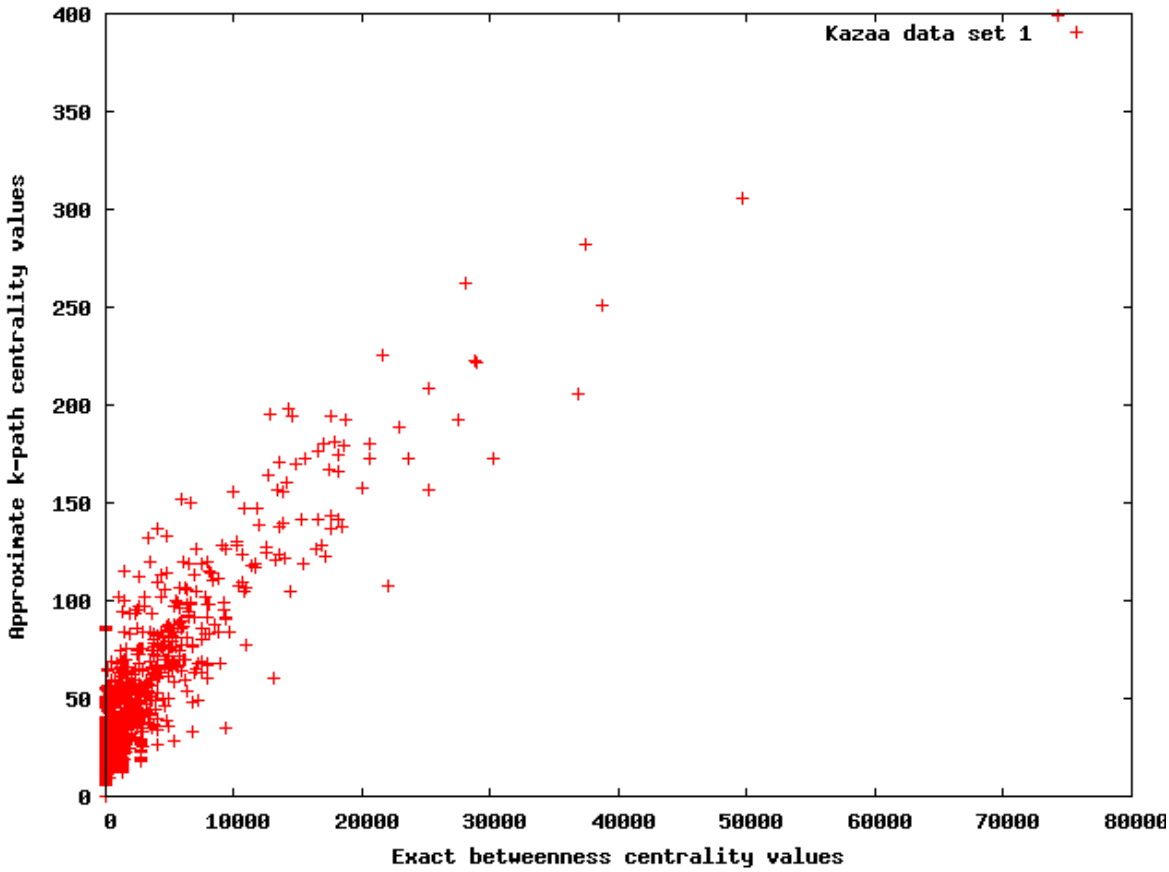

Figure 4.17 Scatter plot for Kazaa data set 1

Table 4.17 Summary information of Kazaa data set 2

Number of vertices (n)	$1,550$
Number of edges (m)	$8,028$
Density ($d = m/\binom{n}{2}$)	0.669%
Directed or undirected	undirected
Weighted or unweighted	weighted
Running time of Brandes' betweenness centrality algorithm	13 seconds
Alpha	0.01
Path length ($\ln(m+n)$)	9
Running time of our k-path centrality algorithm	5 seconds
Correlation coefficient between exact betweenness values and approximate k-path centrality values over all vertices	0.8735
Percentage of vertices common to both the top 5 betweenness values and the top 5 k-path centrality values	60%
Percentage of vertices common to both the top 10 betweenness values and the top 10 k-path centrality values	80%
Percentage of vertices common to both the top 20 betweenness values and the top 20 k-path centrality values	80%
Percentage of vertices common to both the top 50 betweenness values and the top 50 k-path centrality values	78%
Percentage of vertices common to both the top 100 betweenness values and the top 100 k-path centrality values	83%

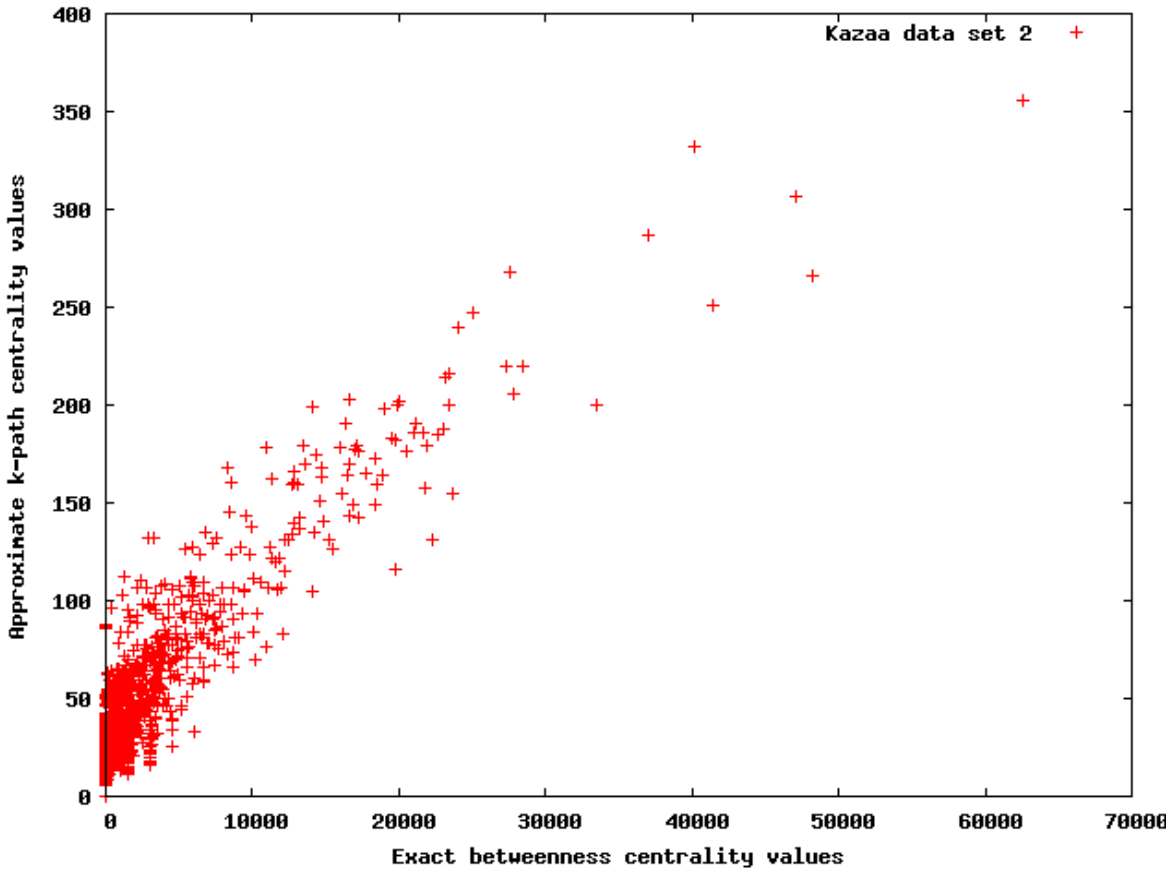

Figure 4.18 Scatter plot for Kazaa data set 2

Table 4.18 Summary information of Kazaa data set 3

Number of vertices (n)	$2,424$
Number of edges (m)	$13,354$
Density ($d = m/\binom{n}{2}$)	0.455%
Directed or undirected	undirected
Weighted or unweighted	weighted
Running time of Brandes' betweenness centrality algorithm	16 seconds
Alpha	0.01
Path length ($\ln(m+n)$)	10
Running time of our k-path centrality algorithm	8 seconds
Correlation coefficient between exact betweenness values and approximate k-path centrality values over all vertices	0.8063
Percentage of vertices common to both the top 5 betweenness values and the top 5 k-path centrality values	80%
Percentage of vertices common to both the top 10 betweenness values and the top 10 k-path centrality values	90%
Percentage of vertices common to both the top 20 betweenness values and the top 20 k-path centrality values	85%
Percentage of vertices common to both the top 50 betweenness values and the top 50 k-path centrality values	78%
Percentage of vertices common to both the top 100 betweenness values and the top 100 k-path centrality values	72%

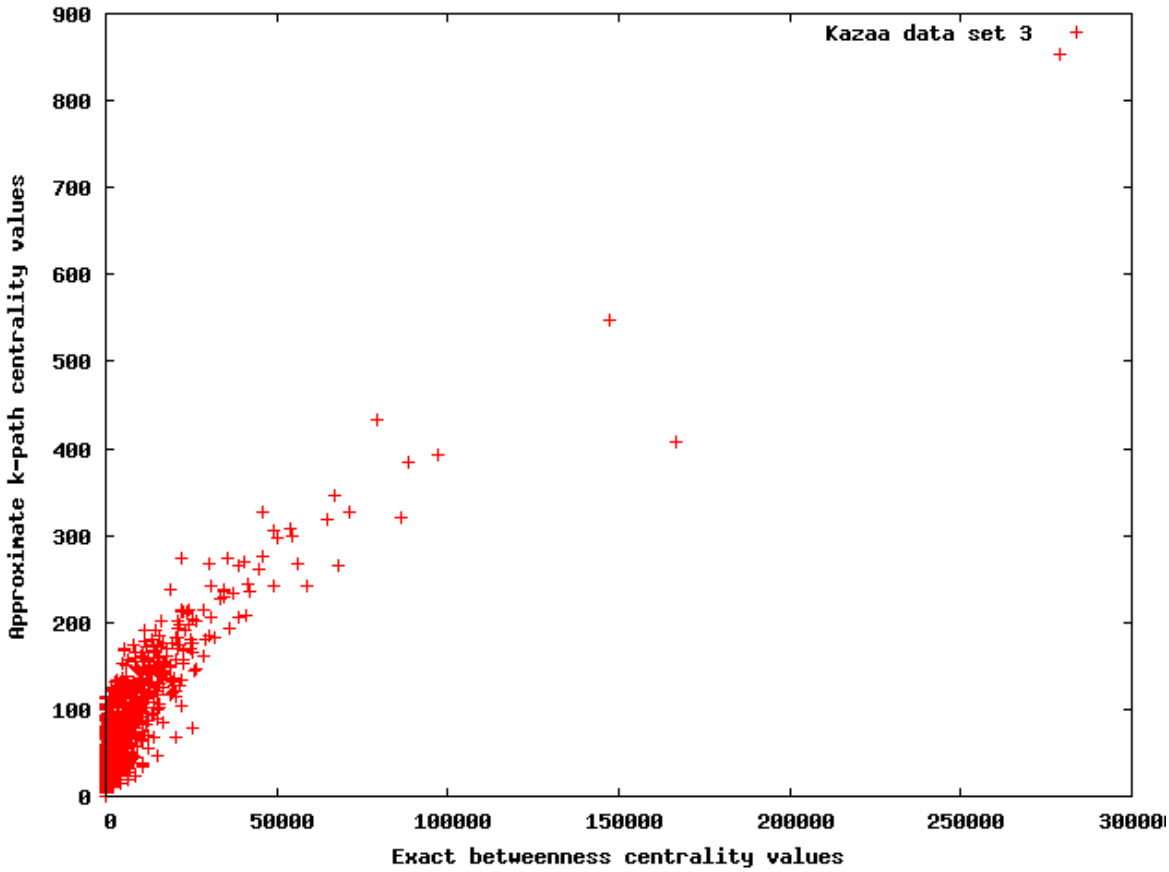

Figure 4.19 Scatter plot for Kazaa data set 3

Table 4.19 Summary information of Computational Geometry data set

Number of vertices (n)	$6,158$
Number of edges (m)	$11,898$
Density ($d = m/\binom{n}{2}$)	0.063%
Directed or undirected	undirected
Weighted or unweighted	weighted
Running time of Brandes' betweenness centrality algorithm	42 seconds
Alpha	0.01
Path length ($\ln(m+n)$)	10
Running time of our k-path centrality algorithm	9 seconds
Correlation coefficient between exact betweenness values and approximate k-path centrality values over all vertices	0.8254
Percentage of vertices common to both the top 5 betweenness values and the top 5 k-path centrality values	60%
Percentage of vertices common to both the top 10 betweenness values and the top 10 k-path centrality values	60%
Percentage of vertices common to both the top 20 betweenness values and the top 20 k-path centrality values	65%
Percentage of vertices common to both the top 50 betweenness values and the top 50 k-path centrality values	72%
Percentage of vertices common to both the top 100 betweenness values and the top 100 k-path centrality values	75%

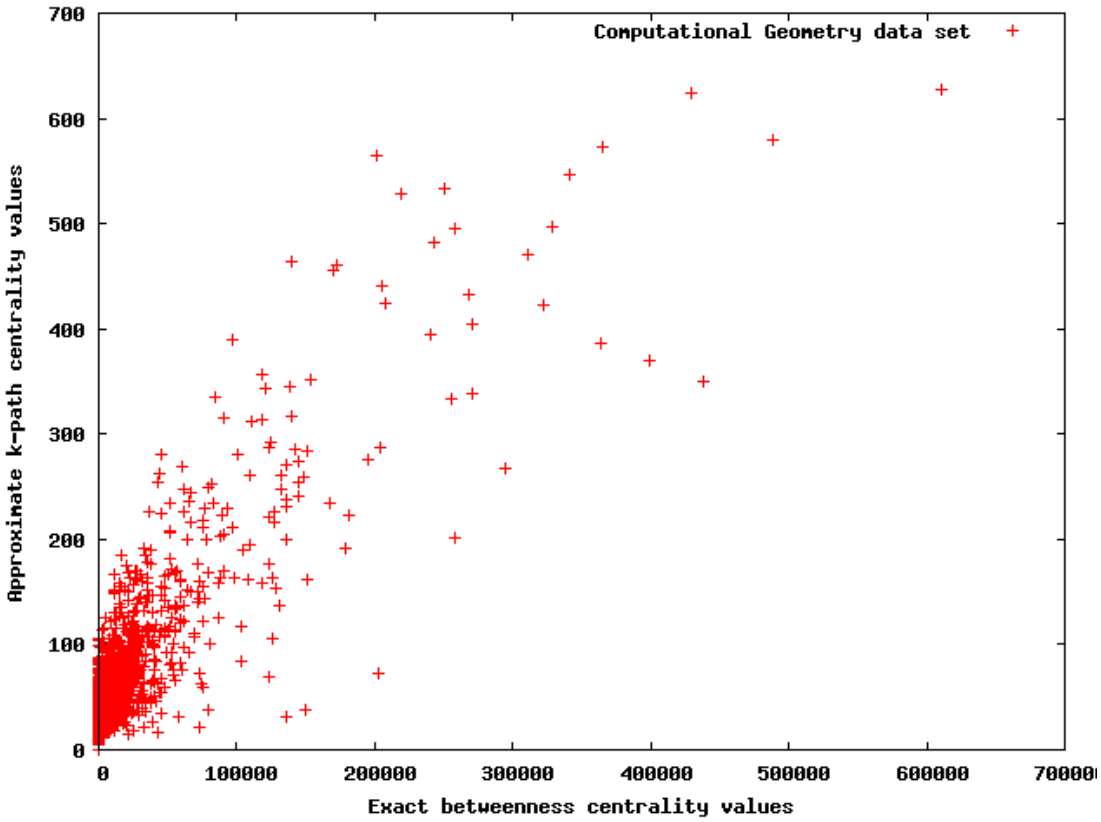

Figure 4.20 Scatter plot for Computational Geometry data set

Table 4.20 Summary information of Small World data set

Number of vertices (n)	233
Number of edges (m)	994
Density ($d = m/\binom{n}{2}$)	3.662%
Directed or undirected	undirected
Weighted or unweighted	unweighted
Running time of Brandes' betweenness centrality algorithm	< 1 seconds
Alpha	0.01
Path length ($\ln(m+n)$)	7
Running time of our k-path centrality algorithm	< 1 seconds
Correlation coefficient between exact betweenness values and approximate k-path centrality values over all vertices	0.8938
Percentage of vertices common to both the top 5 betweenness values and the top 5 k-path centrality values	100%
Percentage of vertices common to both the top 10 betweenness values and the top 10 k-path centrality values	100%
Percentage of vertices common to both the top 20 betweenness values and the top 20 k-path centrality values	80%

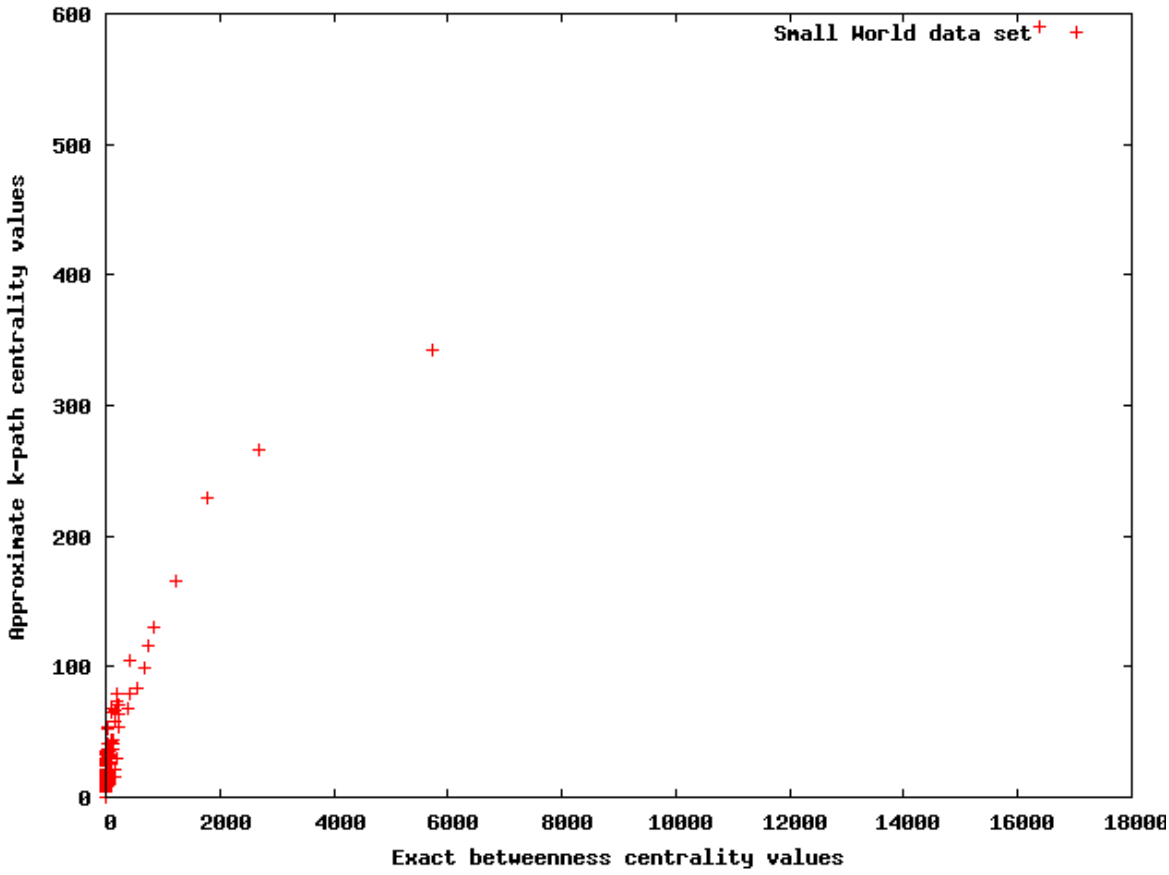

Figure 4.21 Scatter plot for Small World data set

Table 4.21 Summary information of Small, Griffith and Descendants data set

Number of vertices (n)	$1,024$
Number of edges (m)	$4,922$
Density ($d = m/\binom{n}{2}$)	0.455%
Directed or undirected	undirected
Weighted or unweighted	unweighted
Running time of Brandes' betweenness centrality algorithm	3 seconds
Alpha	0.01
Path length ($\ln(m+n)$)	9
Running time of our k-path centrality algorithm	2 seconds
Correlation coefficient between exact betweenness values and approximate k-path centrality values over all vertices	0.9232
Percentage of vertices common to both the top 5 betweenness values and the top 5 k-path centrality values	100%
Percentage of vertices common to both the top 10 betweenness values and the top 10 k-path centrality values	80%
Percentage of vertices common to both the top 20 betweenness values and the top 20 k-path centrality values	90%
Percentage of vertices common to both the top 50 betweenness values and the top 50 k-path centrality values	84%
Percentage of vertices common to both the top 100 betweenness values and the top 100 k-path centrality values	79%

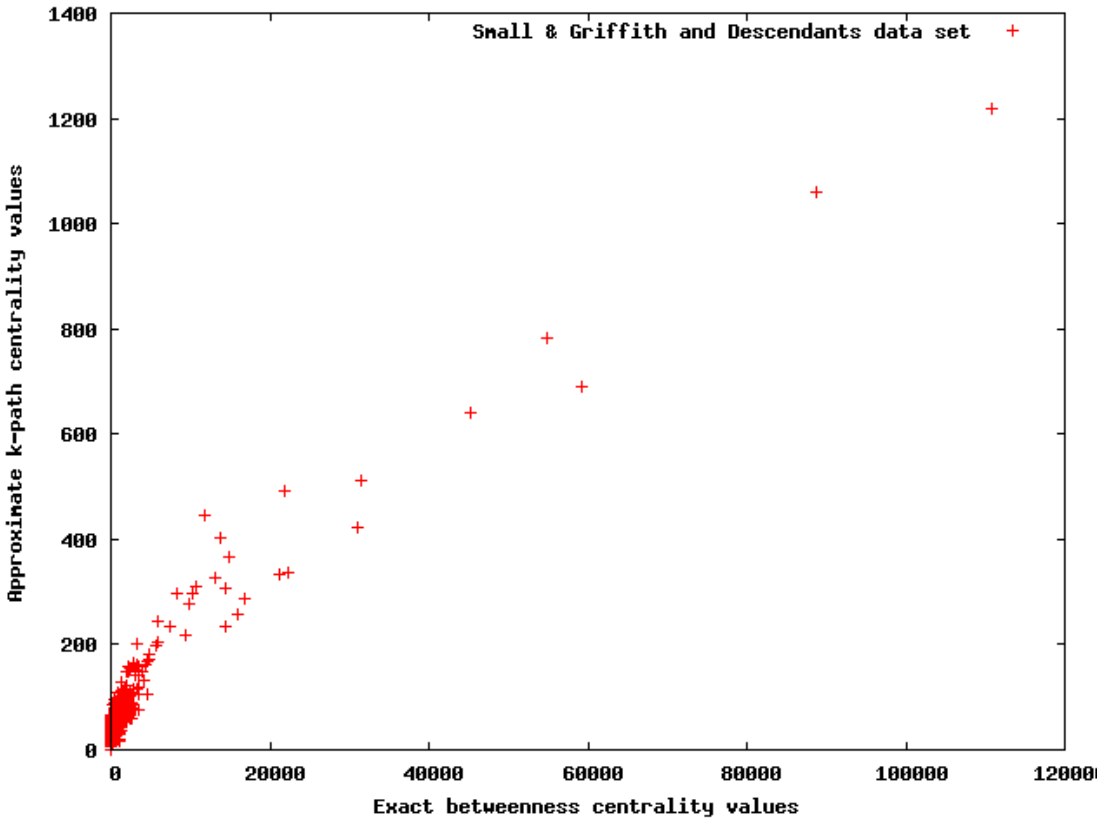

Figure 4.22 Scatter plot for Small, Griffith and Descendants data set

Table 4.22 Summary information of Scientometrics data set

Number of vertices (n)	$2,729$
Number of edges (m)	$10,416$
Density ($d = m/\binom{n}{2}$)	0.280%
Directed or undirected	undirected
Weighted or unweighted	unweighted
Running time of Brandes' betweenness centrality algorithm	22 seconds
Alpha	0.01
Path length ($\ln(m+n)$)	9
Running time of our k-path centrality algorithm	5 seconds
Correlation coefficient between exact betweenness values and approximate k-path centrality values over all vertices	0.8258
Percentage of vertices common to both the top 5 betweenness values and the top 5 k-path centrality values	100%
Percentage of vertices common to both the top 10 betweenness values and the top 10 k-path centrality values	70%
Percentage of vertices common to both the top 20 betweenness values and the top 20 k-path centrality values	90%
Percentage of vertices common to both the top 50 betweenness values and the top 50 k-path centrality values	78%
Percentage of vertices common to both the top 100 betweenness values and the top 100 k-path centrality values	79%

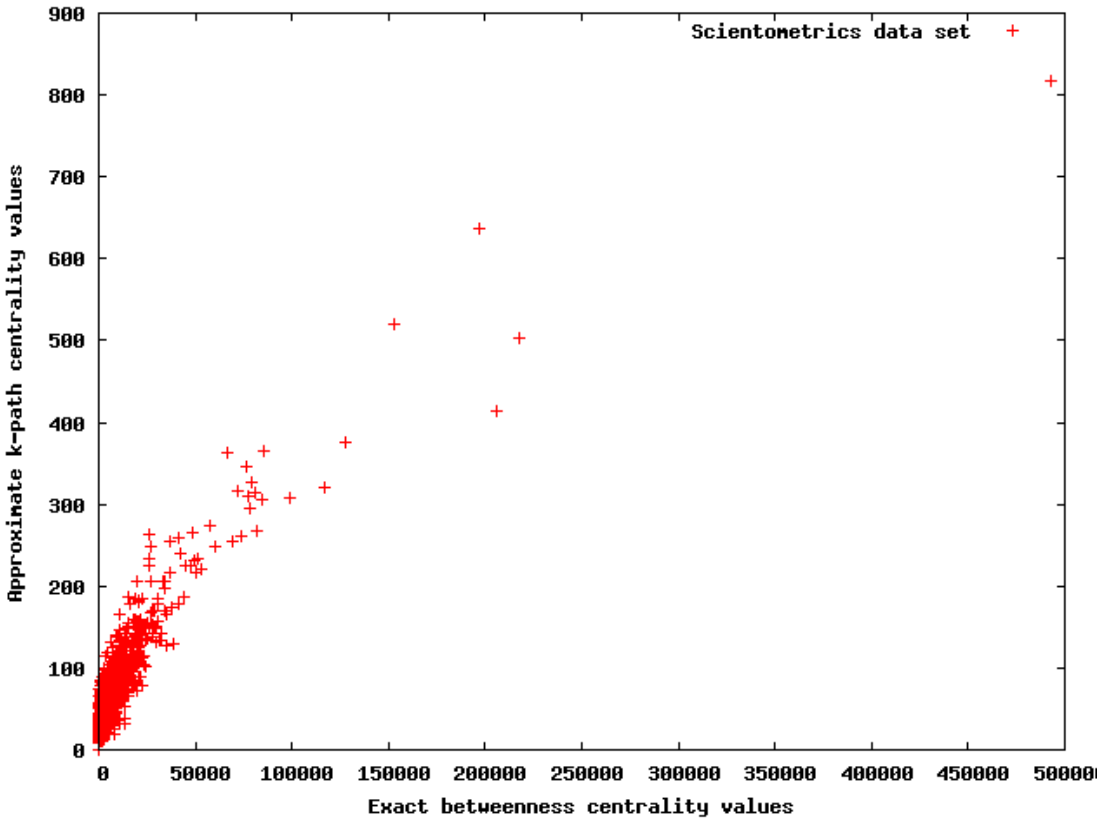

Figure 4.23 Scatter plot for Scientometrics data set

Table 4.23 Summary information of Self-Organizing Maps data set

Number of vertices (n)	$3,772$
Number of edges (m)	$112,731$
Density ($d = m/\binom{n}{2}$)	1.585%
Directed or undirected	undirected
Weighted or unweighted	unweighted
Running time of Brandes' betweenness centrality algorithm	38 seconds
Alpha	0.01
Path length ($\ln(m+n)$)	10
Running time of our k-path centrality algorithm	22 seconds
Correlation coefficient between exact betweenness values and approximate k-path centrality values over all vertices	0.9450
Percentage of vertices common to both the top 5 betweenness values and the top 5 k-path centrality values	100%
Percentage of vertices common to both the top 10 betweenness values and the top 10 k-path centrality values	90%
Percentage of vertices common to both the top 20 betweenness values and the top 20 k-path centrality values	85%
Percentage of vertices common to both the top 50 betweenness values and the top 50 k-path centrality values	80%
Percentage of vertices common to both the top 100 betweenness values and the top 100 k-path centrality values	77%

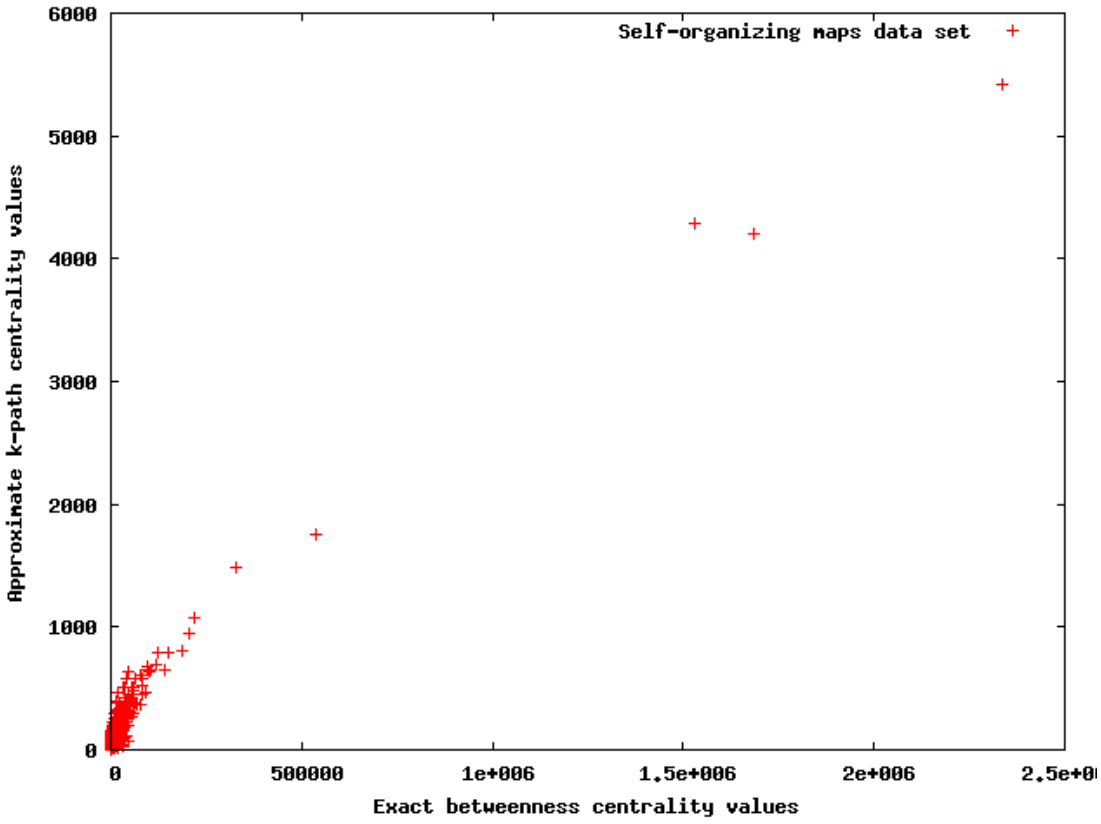

Figure 4.24 Scatter plot for Self-Organizing Maps data set

Table 4.24 Summary information of Zewail data set

Number of vertices (n)	$6,651$
Number of edges (m)	$54,253$
Density ($d = m/\binom{n}{2}$)	0.245%
Directed or undirected	undirected
Weighted or unweighted	unweighted
Running time of Brandes' betweenness centrality algorithm	155 seconds
Alpha	0.01
Path length ($\ln(m+n)$)	11
Running time of our k-path centrality algorithm	30 seconds
Correlation coefficient between exact betweenness values and approximate k-path centrality values over all vertices	0.6676
Percentage of vertices common to both the top 5 betweenness values and the top 5 k-path centrality values	40%
Percentage of vertices common to both the top 10 betweenness values and the top 10 k-path centrality values	70%
Percentage of vertices common to both the top 20 betweenness values and the top 20 k-path centrality values	70%
Percentage of vertices common to both the top 50 betweenness values and the top 50 k-path centrality values	78%
Percentage of vertices common to both the top 100 betweenness values and the top 100 k-path centrality values	79%

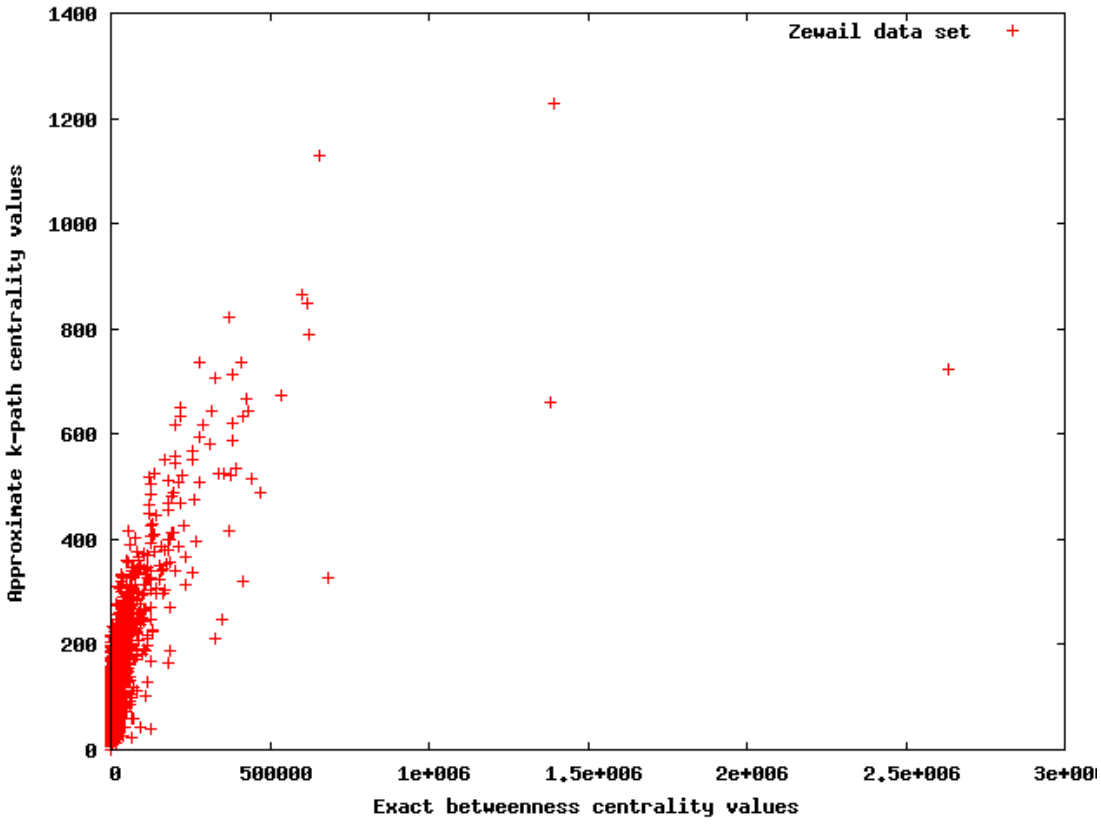

Figure 4.25 Scatter plot for Zewail data set

CHAPTER 5

CONCLUSION

In this thesis, we introduced a new centrality measure called k-path centrality for unweighted graphs. Let n denote the number of vertices and m denote the number of edges in an input graph. We gave a polynomial-time randomized algorithm that, on any input graph, allows us to distinguish between vertices that have high k-path centrality values and vertices that have low k-path centrality values. This algorithm has error probability at most $1/n^2$ and runs in time $\mathcal{O}(k^2\epsilon^{-2}n^{1-\alpha}\ln n)$, where $\alpha \in (0,1)$, $\epsilon \in (0,1/2)$ and integer $k \in [1,n]$ are adjustable parameters. Then we extended our definition of k-path centrality to weighted graphs and gave similar polynomial-time randomized algorithm that distinguishes between high k-path centrality vertices and low k-path centrality vertices in weighted graphs. We showed the analysis of the randomized algorithm for unweighted graphs, and that the analysis of the randomized algorithm for weighted graphs is similar.

Next, we presented a polynomial-time randomized approximation algorithm for computing the k-path centrality values of all vertices in any input graph. This algorithm has an additive error at most $n^{1/2+\alpha}$ and runs in time $O(k^3n^{1-2\alpha}\ln n)$, where $\alpha \in (0,1/2)$ and integer $k \in [1,n]$ are adjustable parameters. Through extensive experimentations on both real and randomly generated networks, we illustrated that vertices with high betweenness centrality values also have high k-path centrality values. Scatter plots between exact betweenness centrality values and approximate k-path centrality values indicate a linear correlation and high correlation coefficient. In all our experiments, we set k to $\ln(n+m)$ and α to 0.01. With these choices of parameters, our randomized approximation algorithm for k-path centrality is asymptotically faster than Brandes' betweenness centrality algorithm. We compared the running time of these algorithms on both real and randomly generated networks. By experimentation, we

demonstrated that our randomized approximation algorithm for k-path centrality is much faster than Brandes' betweenness centrality algorithm.

In this thesis, we did not present any formal justification on why we choose α as 0.01 and k as $\ln(n+m)$. Our choice of these parameters was motivated by the quality of our experimental results. In our future work, we would like to find out the optimal values for these parameters through rigorous analysis. We mention in conclusion that our algorithm is extremely efficient (in terms of time and memory usage) in finding vertices that have high k-path centrality values. Moreover, since approximate k-path centrality values computed by our algorithm show a linear correlation with exact betweenness centrality values in our experimental results, the notion of k-path centrality and the algorithms proposed in this thesis may have practical value in the analysis of networks.

REFERENCES

[Ant71] J. Anthonisse. The rush in a directed graph. Technical Report BN9/71, Stichting Mathematisch Centrum, Amsterdam, Netherlands, 1971.

[Bat03] V. Batagelj. Efficient algorithms for citation network analysis. Technical Report cs.DL/0309023v1, arXiv.org e-Print archive, September 2003.

[BCC+03] D. Bu, L. Cai, R. Chen, G. Li, L. Ling, H. Lu, S. Sun, H. Xue, J. Zhang, N. Zhang, Y. Zhao, and X. Zhu. Topological structure analysis of the protein-protein interaction network in budding yeast. *Nucl. Acids Res.*, 31(9):2443–2450, May 2003.

[BE06] S. Borgatti and M. Everett. A graph-theoretic perspective on centrality. *Social Networks*, 28(4):466–484, 2006.

[Bea65] M. Beauchamp. An improved index of centrality. *Behavioral Science*, 10:161–163, 1965.

[BJMO07] A. Barabsi, H. Jeong, S. Mason, and Z. Oltvai. Network databases. http://www.nd.edu/∼networks/resources.htm, 2007.

[BKMM07] D. Bader, S. Kintali, K. Madduri, and M. Mihail. Approximating betweenness centrality. In *Proceedings of the Workshop on Algorithms and Models for the Web-Graph (WAW)*, pages 124–137. SIAM, 2007.

[BM06] V. Batagelj and A. Mrvar. Pajek datasets. http://vlado.fmf.uni-lj.si/pub/networks/data/, 2006.

[Bon72] P. Bonacich. Factoring and weighting approaches to status scores and clique identification. *Journal of Mathematical Sociology*, 2(1):113–120, 1972.

[BP07] U. Brandes and C. Pich. Centrality estimation in large networks. *International Journal of Bifurcation and Chaos*, 17(7):2303–2318, 2007. Special Issue on *Complex Networks Structure and Dynamics*.

[Bra01] U. Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25(2):163–177, 2001.

[Bra08] U. Brandes. On variants of shortest-path betweenness centrality and their generic computation. *Social Networks*, 30(2):136–145, May 2008.

[BRTC08] P. Boas, A. Rodrigues, G. Travieso, and L. Costa. Border trees of complex networks. *Journal of Physics A Mathematical General*, 41(22):224005, June 2008.

[CHL10] S. Cafieri, P. Hansen, and L. Liberti. Edge ratio and community structure in networks. *Physical Review E*, 81(2):026105, Feb 2010.

[DFLJ07] H. Du, M. Feldman, S. Li, and X. Jin. An algorithm for detecting community structure of social networks based on prior knowledge and modularity. *Complexity*, 12:53–60, 2007.

[DH97] T. Davis and Y. Hu. University of Florida sparse matrix collection. http://www.cise.ufl.edu/research/sparse, 1997.

[EW01] D. Eppstein and J. Wang. Fast approximation of centrality. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 228–229. ACM Press, 2001.

[Fre77] L. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977.

[GCZ09] Y. Guo, C. Chen, and S. Zhou. Fingerprint for network topologies. In Jie Zhou, editor, *Complex (2)*, volume 5 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 1666–1677. Springer, 2009.

[GL08] R. Ghosh and K. Lerman. Community detection using a measure of global influence. Technical Report cs.CY/0805.4606v1, arXiv.org e-Print archive, August 2008.

[GSS08] R. Geisberger, P. Sanders, and D. Schultes. Better approximation of betweenness centrality. In *Proceedings of the 9th Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 3–12. SIAM, 2008.

[HH95] P. Hage and F. Harary. Eccentricity and centrality in networks. *Social Networks*, 17:57–63, 1995.

[HJ08] B. Huang and T. Jebara. Maximum likelihood graph structure estimation with degree distributions. In *Analyzing Graphs: Theory and Applications, NIPS Workshop*, 2008.

[IRF04] A. Iamnitchi, M. Ripeanu, and I. Foster. Small-world file-sharing communities. In *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 952–963, March 2004.

[JGH10] D. Jansen, R. Gortz, and R. Heidler. Knowledge production and the structure of collaboration networks in two scientific fields. *Scientometrics*, 83:219–241, 2010.

[Knu93] D. Knuth. *The Stanford GraphBase: A Platform for Combinatorial Computing*. Addison-Wesley, 1993.

[LN05] D. Liben-Nowell. *An Algorithmic Approach to Social Networks*. Ph.D. thesis, Massachusetts Institute of Technology, Electrical Engineering and Computer Science Department, June 2005.

[MG09] I. Morarescu and A. Girard. Opinion dynamics with decaying confidence: Application to community detection in graphs. Technical Report math.OC/0911.5239v1, arXiv.org e-Print archive, Nov 2009.

[MMO09] P. McSweeney, K. Mehrotra, and J. Oh. A new community detection algorithm based on makov-chains and a team formation model. In *Proceedings of the 2009 International Conference on Advances in Social Network Analysis and Mining (ASONAM)*, pages 371–372. IEEE Computer Society, July 2009.

[New01a] M. Newman. Scientific collaboration networks. I. network construction and fundamental results. *Physical Review E*, 64(1):016131, June 2001.

[New01b] M. Newman. The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences of the United States of America*, 98(2):404–409, January 2001.

[New01c] M. Newman. Who is the best connected scientist? a study of scientific coauthorship networks. *Physical Review E*, 64:016132, 2001.

[New04] M. Newman. Detecting community structure in networks. *The European Physical Journal B - Condensed Matter and Complex Systems*, 38:321–330, March 2004.

[New06] M. Newman. Finding community structure in networks using the eigenvectors of matrices. *Arxiv preprint physics/0605087*, 2006.

[New08] M. Newman. Network data. http://www-personal.umich.edu/∼mejn/netdata/, 2008.

[NL07] M. Newman and E. Leicht. Mixture models and exploratory analysis in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 104(23):9564–9569, June 2007.

[PPZ10] M. Piraveenan, M. Prokopenko, and A. Zomaya. Local assortativeness in scale-free networks. *EPL (Europhysics Letters)*, 89(4):49901, 2010.

[Res10] Research Computing at University of South Florida. Circe hardware. https://rc.usf.edu/trac/doc/wiki/CirceHardware, 2010.

[Sab66] G. Sabidussi. The centrality index of a graph. *Psychometrika*, 31:581–603, 1966.

[Sco00] P. Scott. *Social Network Analysis: A Handbook*. Sage Publications Ltd., 2000.

[Shi53] A. Shimbel. Structural parameters of communication networks. *Bulletin of Mathematical Biophysics*, 15:501–507, 1953.

[WF94] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994.

[Zac77] W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33:452–473, 1977.

ABOUT THE AUTHOR

Tharaka Alahakoon received his B. S. in Computer Science from the University of South Florida (USF). He is currently pursuing two Master's degrees from USF, one in Computer Science and the other in Mathematics. His research interests are Randomized Algorithms, Combinatorics and Graph Theory. He is conducting his research work under Dr. Rahul Tripathi in the Department of Computer Science and Engineering at USF. In addition to his research work, he is currently working as a graduate teaching assistant in the Department of Mathematics and Statistics at USF.