TiCTak: Target-Specific Centrality Manipulation on Large Networks

by

Ruiyue Peng

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved November 2016 by the
Graduate Supervisory Committee:

Hanghang Tong, Chair
Jingrui He
Hasan Davulcu

ARIZONA STATE UNIVERSITY

December 2016

ABSTRACT

Measuring node centrality is a critical common denominator behind many important graph mining tasks. While the existing literature offers a wealth of different node centrality measures, it remains a daunting task on how to intervene the node centrality in a desired way. In this thesis, I study the problem of minimizing the centrality of one or more target nodes by edge operation. The heart of the proposed method is an accurate and efficient algorithm to estimate the impact of edge deletion on the spectrum of the underlying network, based on the observation that the edge deletion is essentially a local, sparse perturbation to the original network. Extensive experiments are conducted on a diverse set of real networks to demonstrate the effectiveness, efficiency and scalability of our approach. In particular, it is average of 260.95%, in terms of minimizing eigen-centrality, better than the standard matrix-perturbation based algorithm, with lower time complexity.

TABLE OF CONTENTS

Page

LIST OF TABLES iv

LIST OF FIGURES v

CHAPTER

1 INTRODUCTION 1

2 PROBLEM DEFINITIONS 5

2.1 Eigen-Centrality 6

2.2 Related Algorithms for Updating Eigenpairs 6

2.2.1 Standard Matrix-Perturbation Theory 7

2.2.2 Cheetah 8

3 ALGORITHM AND ANALYSIS 10

3.1 A Proposed Algorithm for Updating Eigen-Centrality 10

3.2 Proposed TiCTak and TiCTak+ Algorithm 14

4 EXPERIMENTAL EVALUATIONS 17

4.1 Experimental Setup 17

4.2 Effectiveness of TiCTak and TiCTak+ Algorithms 19

4.3 Efficiency of TiCTak and TiCTak+ Algorithms 25

5 SOFTWARE PACKAGES 29

5.1 GCO_Melt_Nodes 29

5.2 GCO_Melt_Edges 30

5.3 GCO_TiCTak_Edges 33

6 RELATED WORK 34

7 CONCLUSION 36

REFERENCES 38

APPENDIX Page

A PROOF OF FORMULATIONS 42

LIST OF TABLES

Table Page

1 Notation Used in Text. 5

2 Updating Eigen-Centrality Algorithms 13

3 The Summary of Different Algorithms 17

4 Datasets 18

5 'Method' of GCO_Melt_Nodes 30

6 'Method' of GCO_Melt_Edges 31

7 'Method' of GCO_TiCTak_Edges 33

LIST OF FIGURES

Figure Page

1 A Graph with Target Nodes 1

2 Effectiveness of TiCTak and SMPT Algorithms for LinkedIn Dataset 3

3 A Ten-Node Graph with $\lambda_1 = \lambda_2$ 7

4 Work-Flow of TiCTak vs Work-Flow of TiCTak+ 16

5 Effectiveness of TiCTak, TiCTak+ and Other Algorithms for All Experimental Datasets 20

5 Effectiveness of TiCTak, TiCTak+ and Other Algorithms for All Experimental Datasets 21

5 Effectiveness of TiCTak, TiCTak+ and Other Algorithms for All Experimental Datasets 22

5 Effectiveness of TiCTak, TiCTak+ and Other Algorithms for All Experimental Datasets 23

5 Effectiveness of TiCTak, TiCTak+ and Other Algorithms for All Experimental Datasets 24

6 Effectiveness of TiCTak, TiCTak+ and Other Algorithms for Facebook Dataset 25

7 Efficiency of TiCTak, TiCTak+ and Other Algorithms for Facebook Dataset . 26

8 Scalability of TiCTak and TiCTak+ Algorithms 27

Chapter 1

INTRODUCTION

Node centrality is an indicator of node importance. Measuring node centrality is a critical common denominator behind many important graph mining tasks, e.g., ranking, meme (e.g., virus, idea, rumor, etc) dissemination, recommendation, etc. As such, the research community has developed a wealth of different centrality measures, ranging from eigenvector centrality (Newman 2008), shortest path based centrality (Freeman 1977), PageRank (Page et al. 1999), HITS (Kleinberg 1999), shield value (Tong et al. 2010), to random walks based centrality (Newman 2005; Kang et al. 2011), etc.

While the existing literature offers powerful measures to observe the importance of the nodes, it remains a daunting task on how to intervene the node centrality in a desired way. How to take a shift from observation to intervention for node centrality? Figure 1 shows a graph with target nodes.

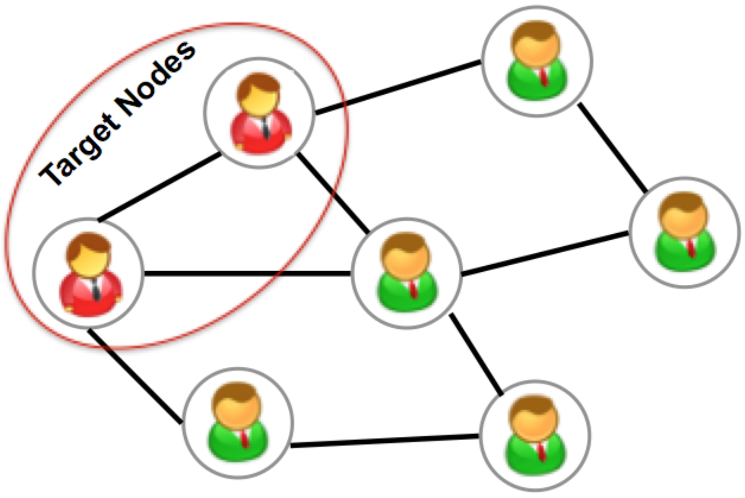


Figure 1: A Graph with Target Nodes

As the first step toward this long-term goal, in this thesis, I study the problem of minimizing the eigen-centrality of a set of target nodes by edge deletion operation. The applications are numerous, such as to neutralize the propaganda message by radical elements (e.g., ISIS) on Internet, to combat link farms on social network sites, etc.

The major hurdle to intervene the node centrality lies in the computation. Suppose there is a network with n nodes and m links, or called edges. In order to find an optimal edge to delete so as to minimize the eigen-centrality, it would take $O(m^2)$ time with the exact algorithm, which is not scalable for large networks. An alternative algorithm is to resort to matrix perturbation theory to approximately estimate the impact of an edge deletion operation on the centrality of the target nodes. For instance, with the first-order matrix perturbation, the time complexity can be reduced to $O(nt^2)$ (Chan, Akoglu, and Tong 2014). The latter has much less complexity than the former because t is far less than m and n.

However, the approximation quality of matrix perturbation theory could quickly be deteriorated, if not collapsed at all, especially when the input networks have small eigen-gaps. This is an inherent limitation, which cannot be overcome by the standard matrix perturbation theory based algorithms ($SMPT$), even if I resort to its higher order variants (Chen and Tong 2016). Therefore, the key challenge lies in how to balance between the approximation quality and the computational time.

In this thesis, accurate and efficient algorithms are proposed to estimate the impact of edge deletion on the spectrum of the underlying network, based on the observation that the edge deletion is essentially a local, sparse perturbation to the original network. Without incurring any additional approximation error (Figure 2), our algorithm has less complexity than $SMPT$. (Chan, Akoglu, and Tong 2014).

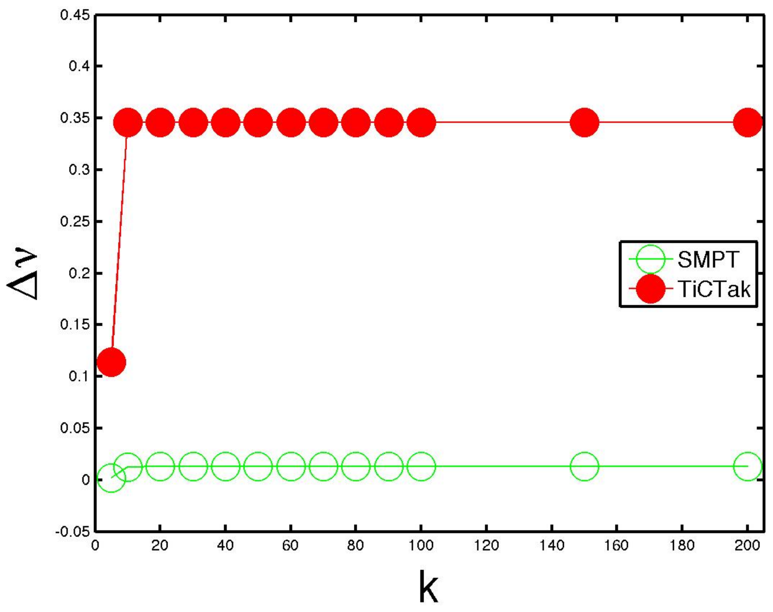


Figure 2: Effectiveness of TiCTak and SMPT Algorithms for LinkedIn Dataset

The main contributions of the thesis are:

Problem Definitions. The intervened aspect of node centrality is defined, i.e., to manipulate the centrality for target nodes by optimizing its underlying network structure.

Algorithm and Analysis. Accurate and efficient algorithms are proposed to select edges to minimize the eigen-centrality, with a linear time complexity.

Experimental Evaluations. Extensive evaluations are conducted on a diverse set of real networks, which consistently demonstrate the effectiveness, efficiency and scalability of the proposed algorithm. It is average of 260.95%, in terms of minimizing eigen-centrality, better than $SMPT$, with lower runtime.

The rest of thesis is organized as follows. The problem is defined in Chapter 2. Algorithms are proposed in Chapter 3. Empirical evaluation is conducted in Chapter 4.

Chapter 5 reviews the related work while Chapter 6 introduces the software packages. Chapter 7 concludes the thesis.

Chapter 2

PROBLEM DEFINITIONS

This thesis addresses the problem of minimizing eigen-centrality of target nodes by deleting edges. The problem can be defined as follows:

Problem: Selecting and deleting edges to minimize eigen-centrality of target nodes

Given: An n x n graph $\mathbf{A}$, a budget of k edges and a set $\mathcal{I}$ of target nodes

Output: A new n x n graph $\widetilde{\mathbf{A}}$ with a set $\mathcal{S}$ of k edges deleted which leads to the largest decrease of eigen-centrality for the target nodes.

Table 1: Notation used in text.

Symbol	Definition and Description
n	the number of the nodes in the graph
m	the number of the edges in the graph
k	the budget (i.e., the specified number of deleted or added edges)
t	the rank of $\mathbf{A}$
ν	eigen-centrality of target nodes
$\mathbf{A}$,$\mathbf{B}$,...	matrix (bold upper case)
$\mathbf{A}(i,j)$	the element at the i^{th} row and j^{th} column of $\mathbf{A}$
$\mathbf{A}(i,:)$	the i^{th} row of matrix $\mathbf{A}$
$\mathbf{A}(:,j)$	the j^{th} column of matrix $\mathbf{A}$
$\mathbf{A}'$	transpose of matrix $\mathbf{A}$
$\Delta\mathbf{A}$	perturbation of $\mathbf{A}$
$\mathbf{a}$,$\mathbf{b}$...	vectors
$\mathcal{I},\mathcal{S}$...	sets (calligraphic)
$\Lambda(j,j)$	the j^{th} eigenvalue of $\mathbf{A}$
$\mathbf{U}(i,j)$	the i^{th} element of j^{th} eigenvector of $\mathbf{A}$
$\Delta\Lambda$	the eigenvalue matrix of $\Delta\mathbf{A}$
$\Delta\mathbf{U}$	the eigenvector matrix of $\Delta\mathbf{A}$

Table 1 lists the notation that is used throughout this thesis. A graph is described by its adjacency matrix. Following the standard notation, bold upper-case is used for matrix (e.g., $\mathbf{A}$). The transpose of matrix $\mathbf{A}$ is represented as $\mathbf{A}'$. Also, the elements are represented in a matrix using a convention similar to $Matlab$, e.g., $\mathbf{A}(i,j)$ is the element at the i^{th} row and the j^{th} column of matrix $\mathbf{A}$, and $\mathbf{A}(:,j)$ is the j^{th} column of matrix $\mathbf{A}$, etc.

2.1 Eigen-Centrality

Based on $Perron{-}Frobenius\ Theory$, eigen-centrality is determined by computing an eigenvector of the adjacency matrix (Newman 2008). Let $\Lambda(1,1)$ be the largest eigenvalue of adjacency matrix $\mathbf{A}$. The eigen-centrality of the nodes of the graph $\mathbf{A}$ is the elements of eigenvector $\mathbf{U}(:,1)$ corresponding to the largest eigenvalue $\Lambda(1,1)$.

Given a set $\mathcal{I}$ of target nodes, the eigen-centrality vector of $\mathcal{I}$ is $\mathbf{U}(\mathcal{I},1)$. The sum of the elements of $\mathbf{U}(\mathcal{I},1)$ is the eigen-centrality of $\mathcal{I}$, which is

$$\nu = \sum_{i \epsilon \mathcal{I}} \mathbf{U}(i,1) \tag{2.1}$$

2.2 Related Algorithms for Updating Eigenpairs

To measure each edge's impact on the eigen-centrality, eigenpairs need to be updated each time after an edge is removed. This section introduces two algorithms for updating eigenpairs: standard matrix-perturbation theory (Chan, Akoglu, and Tong 2014) and *Cheetah* (Li et al. 2015).

2.2.1 Standard Matrix-Perturbation Theory

In (Chan, Akoglu, and Tong 2014), the authors proposed standard matrix-perturbation theory to approximately update eigenpairs. Let $\Delta\mathbf{A}$ be an n x n perturbation matrix to $\mathbf{A}$, where $\Delta\mathbf{A}(p,r) = \Delta\mathbf{A}(r,p) =$ -1 and 0 elsewhere. The eigenvalues of $\Delta\mathbf{A}$ can be updated by

$$\Delta\Lambda(j,j) = -2\mathbf{U}(p,j)\mathbf{U}(r,j) \tag{2.2}$$

The eigenvectors can be updated by

$$\Delta\mathbf{U}(:,j) = \sum_{i=1,i\neq j}^{n} \left(\frac{\mathbf{U}(:,j)'\Delta\mathbf{A}\mathbf{U}(:,j)\mathbf{U}(:,i)}{\Lambda(j,j) - \Lambda(i,i)}\right) \tag{2.3}$$

The time complexity is $O(nt^2)$. Although fast, this is an approximate algorithm and cannot handle the case of $\Lambda(i,i) = \Lambda(j,j)$. It is trivial to show the graph in Figure 3 has $\Lambda(1,1) = \Lambda(2,2) = 1.85$. In this case, this algorithm cannot find $\Delta\mathbf{U}$(:,1) by (2.3).

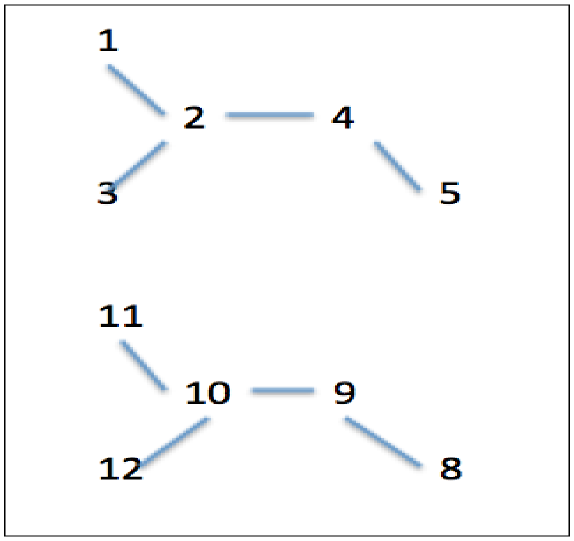


Figure 3: A ten-node graph with $\lambda_1 = \lambda_2$

2.2.2 Cheetah

(Li et al. 2015) proposed another update eigenpairs algorithm, which does not have a constraint and avoids introducing the approximation during the updating process. The key idea of algorithm is the following.

Firstly, obtaining the eigenpair matrices of the $\Delta\mathbf{A}$.

$$\Delta\mathbf{A} = \Delta\mathbf{U}\Delta\Lambda\Delta\mathbf{U}' \tag{2.4}$$

Secondly, performing a partial QR decomposition on the block matrix $[\mathbf{U}\ \Delta\mathbf{U}]$.

$$\begin{bmatrix} \mathbf{U} & \Delta\mathbf{U} \end{bmatrix} = \mathbf{QR} \tag{2.5}$$

where

$$\mathbf{Q} = \begin{bmatrix} \mathbf{U} & \frac{\mathbf{q}_1}{||\mathbf{q}_1||} & \frac{\mathbf{q}_2}{||\mathbf{q}_2||} \end{bmatrix} \tag{2.6}$$

and

$$\mathbf{R} = \begin{bmatrix} \mathbf{I} & \mathbf{r}_1 & \mathbf{r}_2 \\ \mathbf{0} & ||\mathbf{q}_1|| & -||\mathbf{q}_1||\mathbf{r}_1'\mathbf{r}_2 \\ \mathbf{0} & 0 & ||\mathbf{q}_2|| \end{bmatrix} \tag{2.7}$$

Thirdly, constructing a new matrix $\mathbf{Z}$ from the upper triangle matrix $\mathbf{R}$ of the QR decomposition and the eigenvalues matrix of $\mathbf{A}$ and $\Delta\mathbf{A}$.

$$\mathbf{Z} = \mathbf{R}\begin{bmatrix} \Lambda & \mathbf{0} \\ \mathbf{0} & \Delta\Lambda \end{bmatrix}\mathbf{R}' \tag{2.8}$$

Fourthly, obtaining the eigenpair matrices of the $\mathbf{Z}$.

$$\mathbf{Z} = \mathbf{V}\bar{\Lambda}\mathbf{V}' \tag{2.9}$$

$\bar{\Lambda}$ are the new eigenvalue matrix. $\mathbf{Z}$'s eigenvector rotates the orthonormal basis of the QR decomposition to get the new eigenvector matrix $\bar{\mathbf{U}}$.

$$\bar{\mathbf{U}} = \mathbf{QV} \tag{2.10}$$

The time complexity of this algorithm is $O(nt^2)$. Even though *Cheetah* is an accurate method, its time complexity prohibits its use for large networks.

Chapter 3

ALGORITHM AND ANALYSIS

In this chapter, a new algorithm is proposed, called $UdEigen$, to efficiently update eigen-centrality. $TiCTak$ is another algorithm proposed, which used $UdEigen$ as its core module for deletion in order to minimize the eigen-centrality.

3.1 A Proposed Algorithm for Updating Eigen-Centrality

The proposed algorithm is based on *Cheetah* (Li et al. 2015), an algorithm which can accurately updating eigenpairs. I improve it to update eigen-centrality more efficiently.

The eigen-decomposition of $\mathbf{A}$ and $\Delta\mathbf{A}$ are following:

$$\mathbf{A} = \mathbf{U}\Lambda\mathbf{U}' \tag{3.1}$$

$$\Delta\mathbf{A} = \Delta\mathbf{U}\Delta\Lambda\Delta\mathbf{U}' \tag{3.2}$$

where $\mathbf{U}$, Λ are the eigenpair-matrices of $\mathbf{A}$ while $\Delta\mathbf{U}$, $\Delta\Lambda$ are the eigenpair-matrices of $\Delta\mathbf{A}$.

Every time, only an edge (i, j) is deleted from the graph $\mathbf{A}$. Therefore, the perturbation martix $\Delta\mathbf{A}$ is a 2-rank adjacency matrix with $\Delta\mathbf{A}(i, j)$ =$\Delta\mathbf{A}(j, i)$=-1 and 0 elsewhere. The 2-rank eigenvalue matrix $\Delta\Lambda$ and eigenvector matrix $\Delta\mathbf{U}$ of $\Delta\mathbf{A}$ can be calculated. The equation is following:

$$\Delta\Lambda = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{3.3}$$

$\Delta\mathbf{U}$ is an n x 2 matrix with all “0” elements except:

$$\Delta\mathbf{U}(i,1) = \frac{1}{\sqrt{2}}, \Delta\mathbf{U}(j,1) = -\frac{1}{\sqrt{2}} \tag{3.4}$$

$$\Delta\mathbf{U}(i,2) = -\frac{1}{\sqrt{2}}, \Delta\mathbf{U}(j,2) = -\frac{1}{\sqrt{2}}$$

Perform QR decomposition on matrix $[\mathbf{U}\ \Delta\mathbf{U}]$,

$$\begin{bmatrix} \mathbf{U} & \Delta\mathbf{U} \end{bmatrix} = \mathbf{QR} \tag{3.5}$$

where $\mathbf{Q}$ is an n x $(t+2)$ matrix

$$\mathbf{Q} = \begin{bmatrix} \mathbf{U} & \frac{\mathbf{q}_1}{||\mathbf{q}_1||} & \frac{\mathbf{q}_2}{||\mathbf{q}_2||} \end{bmatrix} \tag{3.6}$$

and $\mathbf{R}$ is a $(t+2)$ x $(t+2)$ upper triangle matrix

$$\mathbf{R} = \begin{bmatrix} \mathbf{I} & \mathbf{r}_1 & \mathbf{r}_2 \\ \mathbf{0} & ||\mathbf{q}_1|| & -||\mathbf{q}_1||\mathbf{r}_1'\mathbf{r}_2 \\ \mathbf{0} & 0 & ||\mathbf{q}_2|| \end{bmatrix} \tag{3.7}$$

In (3.7), $\mathbf{I}$ is a t x t identity matrix and column vectors $\mathbf{r}_1$ and $\mathbf{r}_2$ are defined as

$$\mathbf{r}_1 = \mathbf{U}'\Delta\mathbf{U}(:,1) = \frac{1}{\sqrt{2}}(\mathbf{U}(i,:) - \mathbf{U}(j,:))' \tag{3.8}$$

$$\mathbf{r}_2 = \mathbf{U}'\Delta\mathbf{U}(:,2) = -\frac{1}{\sqrt{2}}(\mathbf{U}(i,:) + \mathbf{U}(j,:))' \tag{3.9}$$

In (3.6),

$$\mathbf{q}_1 = \Delta\mathbf{U}(:,1) - \mathbf{U}\mathbf{r}_1 \tag{3.10}$$

$$\mathbf{q}_2 = \Delta\mathbf{U}(:,2) - \mathbf{U}\mathbf{r}_2 + \mathbf{q}_1\mathbf{r}_1'\mathbf{r}_2 \tag{3.11}$$

In Appendix A, I prove that the norm of $\mathbf{q}_1$ and $\mathbf{q}_2$ are

$$||\mathbf{q}_1|| = \sqrt{1 - ||\mathbf{r}_1||^2} \tag{3.12}$$

$$||\mathbf{q}_2|| = \sqrt{1 - ||\mathbf{r}_2||^2 - (\mathbf{r}_1'\mathbf{r}_2)^2(1 + ||\mathbf{r}_1||^2)} \tag{3.13}$$

Now a new (t+2) x (t+2) matrix $\mathbf{Z}$ is constructed from the upper triangle matrix $\mathbf{R}$ of the QR decomposition and the eigenvalue matrices of $\mathbf{A}$ and $\Delta\mathbf{A}$.

$$\mathbf{Z} = \mathbf{R} \begin{bmatrix} \Lambda & \mathbf{0} \\ \mathbf{0} & \Delta\Lambda \end{bmatrix} \mathbf{R}' \tag{3.14}$$

Perform eigen-decomposition of $\mathbf{Z}$

$$\mathbf{Z} = \mathbf{V}\bar{\Lambda}\mathbf{V}' \tag{3.15}$$

where $\bar{\Lambda}$ are the new eigenvalue matrix.

Z's eigenvector rotates the orthonormal basis of the QR decomposition to get the new eigenvectors $\bar{\mathbf{U}}$.

$$\bar{\mathbf{U}} = \mathbf{QV} \tag{3.16}$$

To compute eigen-centrality of target nodes, I only need to calculate $\bar{\mathbf{U}}(\mathcal{I}, 1)$.

$$\bar{\mathbf{U}}(\mathcal{I}, 1) = \mathbf{Q}(\mathcal{I}, :)\mathbf{V}(:, 1) \tag{3.17}$$

where

$$\mathbf{Q}(\mathcal{I}, :) = \begin{bmatrix} \mathbf{U}(\mathcal{I}, :) & \frac{\mathbf{q}_1(\mathcal{I})}{||\mathbf{q}_1||} & \frac{\mathbf{q}_2(\mathcal{I})}{||\mathbf{q}_2||} \end{bmatrix} \tag{3.18}$$

where

$$\mathbf{q}_1(\mathcal{I}) = \Delta\mathbf{U}(\mathcal{I}, 1) - \mathbf{U}(\mathcal{I}, :)\mathbf{r}_1 \tag{3.19}$$

$$\mathbf{q}_2(\mathcal{I}) = \Delta\mathbf{U}(\mathcal{I}, 2) - \mathbf{U}(\mathcal{I}, :)\mathbf{r}_2 + \mathbf{q}_1(\mathcal{I})\mathbf{r}_1'\mathbf{r}_2 \tag{3.20}$$

From (3.17), eigen-centrality is computed

$$\bar{\nu} = \sum_{i \epsilon \mathcal{I}} \bar{\mathbf{U}}(i, 1) \tag{3.21}$$

Algorithm 1 *UdEigen*

```
Input: Eigenvalue matrix $\Lambda$ and eigenvector matrix $\mathbf{U}$ of $\mathbf{A}$, the edge $(i,j)$, and the set of nodes $\mathcal{I}$
Output: Eigen-centrality $\bar{\nu}$
1: Calculate eigenvalue matrix $\Delta\Lambda$ and eigenvector matrix $\Delta\mathbf{U}$ of $\Delta\mathbf{A}$ for edge $(i,j)$ by (3.3,3.4)
2: Partial QR decomposition of $[\mathbf{U}\Delta\mathbf{U}]$
       $\mathbf{Q} = \left[\begin{array}{ccc} \mathbf{U} & \frac{\mathbf{q}_1}{||\mathbf{q}_1||} & \frac{\mathbf{q}_2}{||\mathbf{q}_2||} \end{array}\right]$
       $\mathbf{R} = \left[\begin{array}{ccc} \mathbf{I} & \mathbf{r}_1 & \mathbf{r}_2 \\ \mathbf{0} & ||\mathbf{q}_1|| & -||\mathbf{q}_1||\mathbf{r}_1'\mathbf{r}_2 \\ \mathbf{0} & 0 & ||\mathbf{q}_2|| \end{array}\right]$
3: Set $\mathbf{Z} = \mathbf{R}\left[\begin{array}{cc} \Lambda & \mathbf{0} \\ \mathbf{0} & \Delta\Lambda \end{array}\right]\mathbf{R}'$
4: Eigen-decomposition of $\mathbf{Z} = \mathbf{V}\ \bar{\Lambda}\mathbf{V}'$
5: $\bar{\mathbf{U}}(\mathcal{I},1) = \mathbf{Q}(\mathcal{I},:)\mathbf{V}(:,1)$
6: Calculate eigen-centrality $\bar{\nu} = \sum_{i\epsilon\mathcal{I}} \bar{\mathbf{U}}(i,1)$
7: Return $\bar{\nu}$
```

Algorithm 1 is for updating eigen-centrality after an edge is deleted. From the above description of the algorithm, there is no approximation involved and faster than *Cheetah*. Step 4 has the most computation, which has complexity $O(t^3)$. Table 2 shows time complexity comparison among algorithms for updating eigen-centrality.

Table 2: Updating eigen-centrality algorithms

Algorithm	Time Complexity
UdEigen	$O(t^3)$
Cheetah based	$O(nt^2)$
Standard Matrix-Perturbation based	$O(nt^2)$
Eigen decomposition based	$O(nt + mt + nt^2)$

3.2 Proposed TiCTak and TiCTak+ Algorithm

Now, I proposed an algorithm, called *TiCTak* algorithm, which is able to delete k edges to minimize the eigen-centrality of target nodes as shown in Algorithm 2.

Firstly, perform eigen-decomposition of $\mathbf{A}$

$$\mathbf{A} = \mathbf{U}\Lambda\mathbf{U}' \tag{3.22}$$

Secondly, calculate eigen-centrality of a set $\mathcal{I}$ of target nodes

$$\nu = \sum_{i \epsilon \mathcal{I}} \mathbf{U}(i,1) \tag{3.23}$$

For each edge (i,j), I use *UdEigen* to calculate $\bar{\nu}$

$$\bar{\nu} = UdEigen(\mathbf{U}, \Lambda, i, j, \mathcal{I}) \tag{3.24}$$

Then, I calculate the score of the edge $\Delta\nu$, called eigen-centrality reduction

$$\Delta\nu = \nu - \bar{\nu} \tag{3.25}$$

After scoring all edges, I rank edges according to the scores, choose the biggest k edges as the set $\mathcal{S}$. I delete the top k edges from $\mathbf{A}$. The corresponding new matrix is $\widetilde{\mathbf{A}}$.

Algorithm 2 *TiCTak*

```
Input: Matrix $\mathbf{A}$, the budget of edges $k$ , the rank $t$ and a set $\mathcal{I}$ of nodes
Output: New matrix $\widetilde{\mathbf{A}}$ and the set $\mathcal{S}$ of $k$ edges
 1: $t$ rank eigen decomposition of $\mathbf{A} = \mathbf{U}\Lambda\mathbf{U}'$
 2: Set $n$ is the size of matrix $\mathbf{A}$
 3: Calculate eigen-centrality: $\nu = \sum_{i \epsilon \mathcal{I}} \mathbf{U}(i, 1)$
 4: for each edge $(i, j)$ of $\mathbf{A}$ do
 5:     $\bar{\nu}$=$UdEigen$($\mathbf{U}$,$\Lambda$,$i$,$j$,$\mathcal{I}$)
 6:     Score:$\Delta\nu = \nu - \bar{\nu}$
 7: end for
 8: Find out the set $\mathcal{S}$ of $k$ edges with top $k$ score $\Delta\nu$
 9: Delete the set $\mathcal{S}$ of $k$ edges from $\mathbf{A}$ and get $\widetilde{\mathbf{A}}$
10: Return $\widetilde{\mathbf{A}}$ and the set $\mathcal{S}$ of $k$ edges
```

The time complexity of Step 1 is $O(nt + mt + nt^2)$, $O(1)$ for Step 2, $O(1)$ for Step 3, $O(t^3)$ for Step 5, $O(mt^3)$ for Steps 4 to 9. Therefore, the total time complexity of Algorithm 2 is $O(mt^3)$. Compared with *Cheetah* based algorithm where the time complexity $O(mnt^2)$, Algorithm 2 is much faster.

The space complexity of storing a graph is $O(m)$, $O(t)$ for eigenvaluse, $O(nt)$ for eigenvector, $O(m)$ for scores of all edges, and $O(k)$ for budget nodes. The total space complexity of Algorithm 2 is $O(m + nt + k)$ which is the same to *Cheetah* based algorithm.

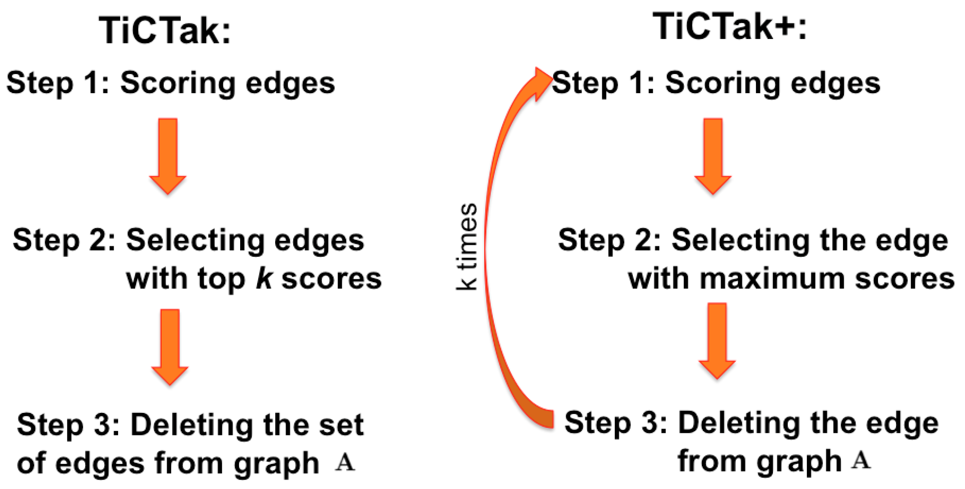

TiCTak:
Step 1: Scoring edges
Step 2: Selecting edges with top *k* scores
Step 3: Deleting the set of edges from graph $\mathbf{A}$
TiCTak+:
Step 1: Scoring edges
Step 2: Selecting the edge with maximum scores
Step 3: Deleting the edge from graph $\mathbf{A}$
k times

Figure 4: Work-flow of TiCTak vs Work-flow of TiCTak+

$TiCTak+$ algorithm is further improvement of $TiCTak$ algorithm. $TiCTak+$ algorithm is $TiCTak$ algorithm with restart. Scoring every edge as before. Only one edge with the maximum score is deleted each time. $\mathbf{A}$ with its eigenpair-matrices $\mathbf{U}$ and Λ are renewed. Scoring, sorting, selecting and deleting an edge with the maximum score totally k times. Finally, return the new graph $\widetilde{\mathbf{A}}$ and the set $\mathcal{S}$ of k edges. The time complexity of $TiCTak+$ is $O(kmt^3)$ and the space complexity is $O(m + nt + k)$. Figure 4 shows the comparison between $TiCTak$ and $TiCTak+$ algorithms.

Chapter 4

EXPERIMENTAL EVALUATIONS

In this chapter, the experimental results are presented for the proposed $TiCTak$ and $TiCTak$+ algorithms. Extensive evaluations are conducted on a diverse set of real networks, which consistently demonstrate the effectiveness, efficiency and scalability of the proposed method. The proposed $TiCTak$ and $TiCTak$+ algorithms are compared with four baseline algorithms. Two of them are standard matrix perturbation theory based algorithms $SMPT$ and $SMPT$+ (Chan, Akoglu, and Tong 2014). The other two are the state-of-the-art melting edge method $NetMelt$ (Tong et al. 2012) and $NetMelt$+ algorithms.

Table 3: The Summary of Different Algorithms

Algorithm	Target-specific	Updating eigen-centrality	Restart	Time complexity
$TiCTak$	Yes	$UdEigen$	No	$O(mt^3)$
$TiCTak$+	Yes	$UdEigen$	Yes	$O(kmt^3)$
$SMPT$	Yes	Standard matrix-pertubation	No	$O(nmt^2)$
$SMPT$+	Yes	Standard matrix-pertubation	Yes	$O(knmt^2)$
$NetMelt$	No	-	No	$O(km+n)$
$NetMelt$+	No	-	Yes	$O(k^2m+kn))$

4.1 Experimental Setup

Nine popular sets of real networks are used for our experiments. Table 4 lists the number of nodes n and edges m of nine real networks to be used for the experiment.

Table 4: Datasets

Dataset	n	m
Router	633	2172
Venue	899	4716
LiveJournal	1000	47641
Facebook	1000	17880
Email	1000	34806
Power	1000	2424
Collaboration	1001	2336
LinkedIn	2000	4468
Airport	2833	5666

All of them are undirected and unweighted graphs. Router network of Oregon Autonomous System is a AS-level connectivity network inferred from Oregon route-views [1]. Facebook network and LinkedIn network are two famous friendship social networks [2]. LiveJournal is a social networking and blogging site with several million members and a large collection of explicit user-defined communities [2]. Enron email communication network covers all the email communication within a dataset of around a half million email accounts [2]. Collaboration network is from the DBLP computer science bibliography [3]. In Collaboration network, two authors are connected if they publish at least one paper together [3]. Power networks represent the topology of the Western States Power Grid of the United States [2].

To achieve fair comparison results, all the experiments were conducted on the same machine running Windows 7 with Intel Core i7-4790 CPU and 32GB memory. All the algorithms are implemented and executed in $MatLab$ simulation environments.

[1]http://topology.eecs.umich.edu/data.html

[2]https://snap.stanford.edu/data/

[3]http://dblp.uni-trier.de/

4.2 Effectiveness of TiCTak and TiCTak+ Algorithms

In this section, the effectiveness of the proposed $TiCTak$ and $TiCTak$+ algorithms along with $SMPT$, $SMPT$+, $NetMelt$ and $NetMelt$+ algorithms are evaluated. The objective of all these algorithms is to minimize eigen-centrality of target nodes. The effectiveness of the algorithms can be measured by the decrease of eigen-centrality $\Delta\nu$ of the graph.

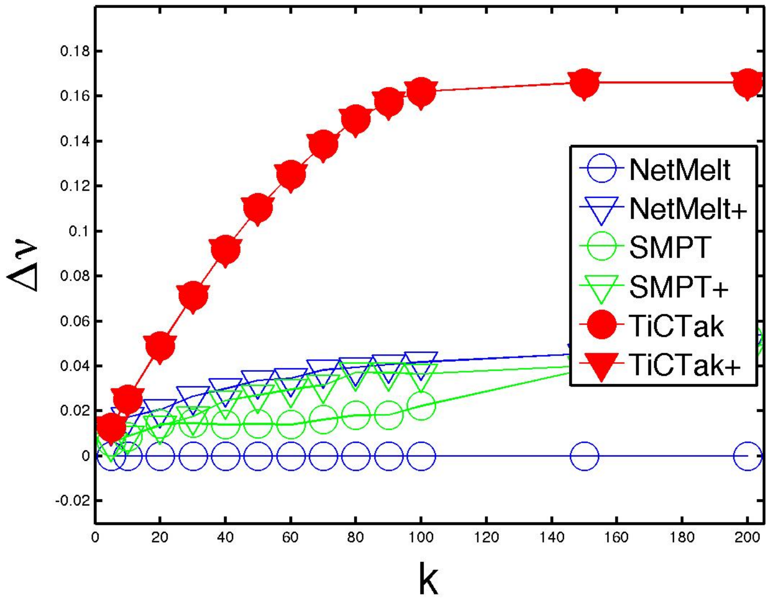
Figure 5(1) to 5(9) shows the effectiveness of $TiCTak$, $TiCTak$+ and other algorithms for all the experimental datasets, where the x-axis represents the budget k and the y-axis represents the reduction of the eigen-centrality. The larger the eigen-centrality decreases, the more effective the algorithm is. All six algorithms are simulated for all nine datasets. It can be observed that the proposed $TiCTak$ and $TiCTak$+ are consistently outperform the rest of algorithms. Both algorithms are 261% on average more effective in terms of eigen-centrality than the best baseline.

Based in Figure 5(1), the experiment results of Facebook dataset, it can be observed that there is no extra gain for most of algorithms after budget k reaches 100 because the curves become flat, which means the eigen-centrality has already been minimized. When the eigen-centrality is minimized, it can be observed that the $TiCTak$ and $TiCTak$+ algorithms result in 338% more reduction of eigen-centrality comparing to other algorithms. The experiment results of Venue dataset, shown in Figure 5(2), are similar to those of results of Facebook dataset.

Based in Figure 5(3), (4), (5) and (6), the experiment results of Linkedin, Collaboration, Airport and Power datasets, I observe that for $TiCTak$ and $TiCTak$+ algorithms, the curves become flat after budget k reaches 20, which means the eigen-centrality is minimized. However, for other algorithms, the curves cannot become flat after budget k reaches 20. For example, in Figure 5(5), the curve for $SMPT$ become flat after budget k reaches 160.

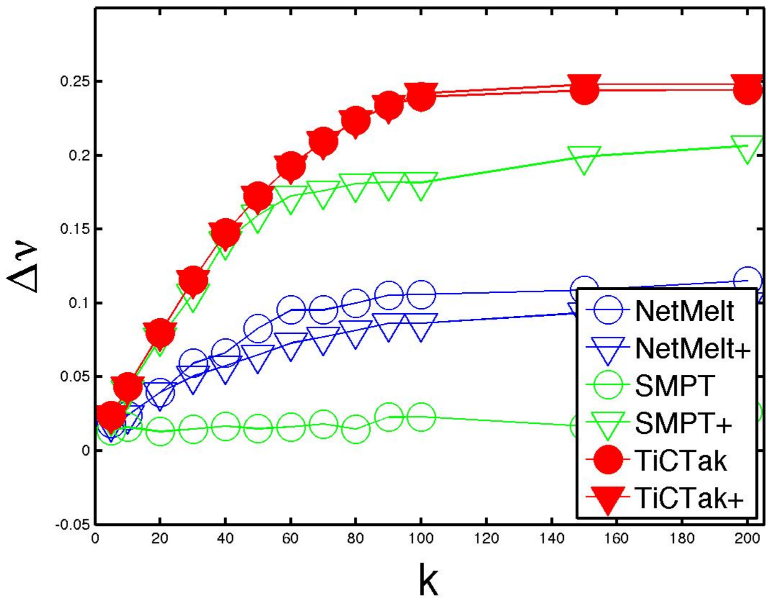
Based in Figure 5(7) and (8), the experiment results of Email and LiveJournal datasets, I observe that for all algorithms, the curves cannot become flat after budget k reaches 200. However, when the budget k reaches 200, it can be observed that the $TiCTak$ and $TiCTak+$ algorithms result in 38% more reduction of eigen-centrality comparing to best baseline algorithm.

$TiCTak$ and $TiCTak+$ have equally good performance except in the Router dataset, shown in Figure 5(9), where $TiCTak+$ leads to the biggest decrease of the eigen-centrality.

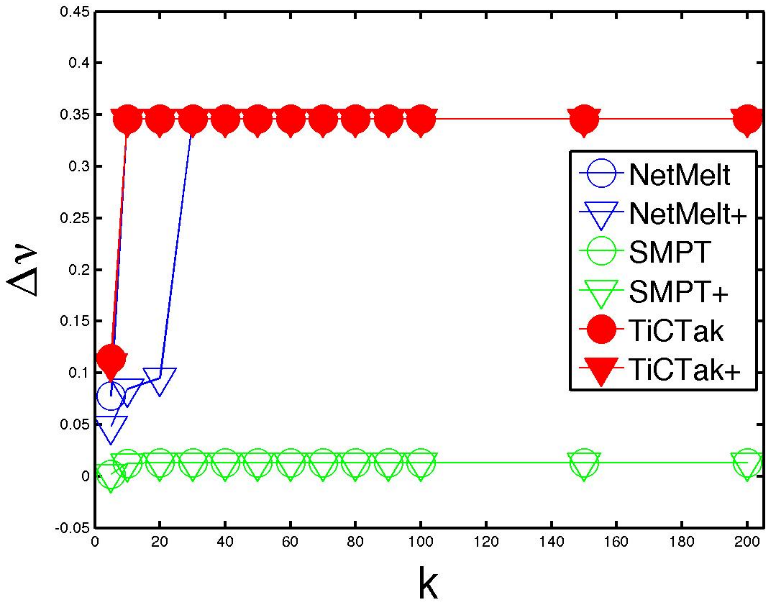


(1) Facebook

Figure 5: Effectiveness of TiCTak, TiCTak+ and Other Algorithms for All Experimental Datasets

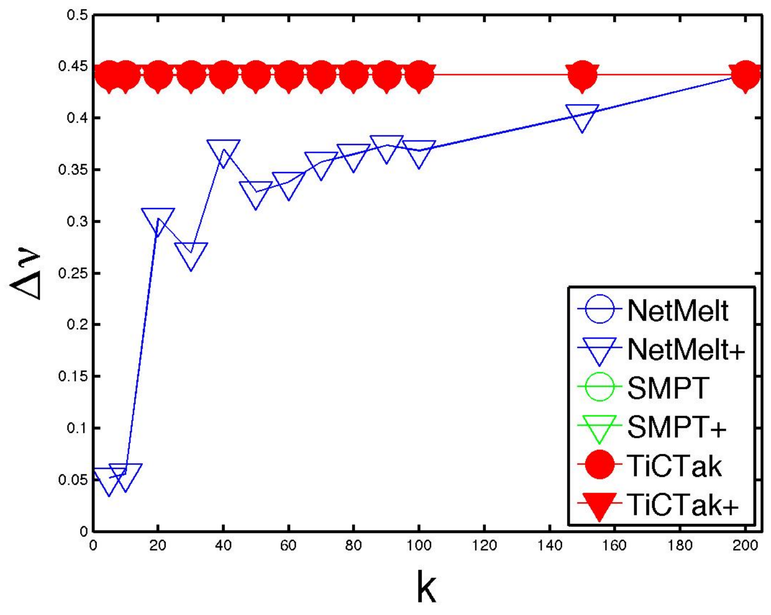


(2) Venue

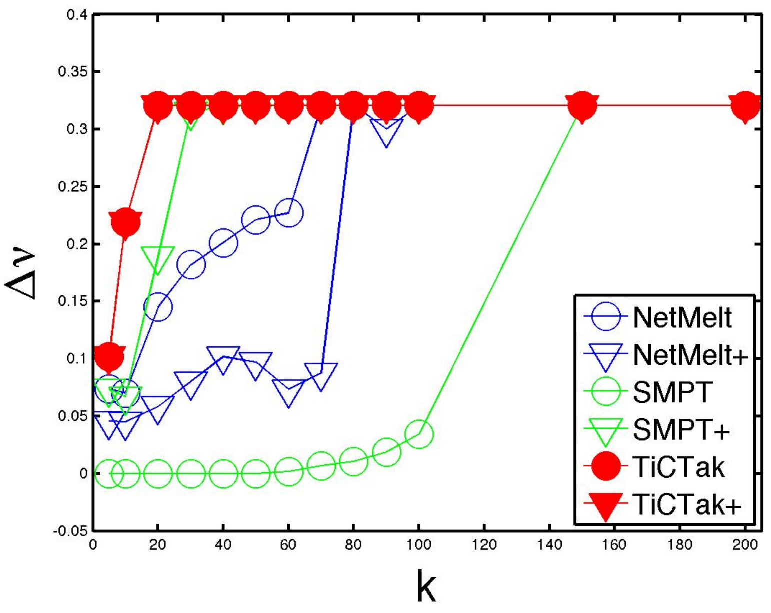


(3) LinkedIn

Figure 5: Effectiveness of TiCTak, TiCTak+ and Other Algorithms for All Experimental Datasets



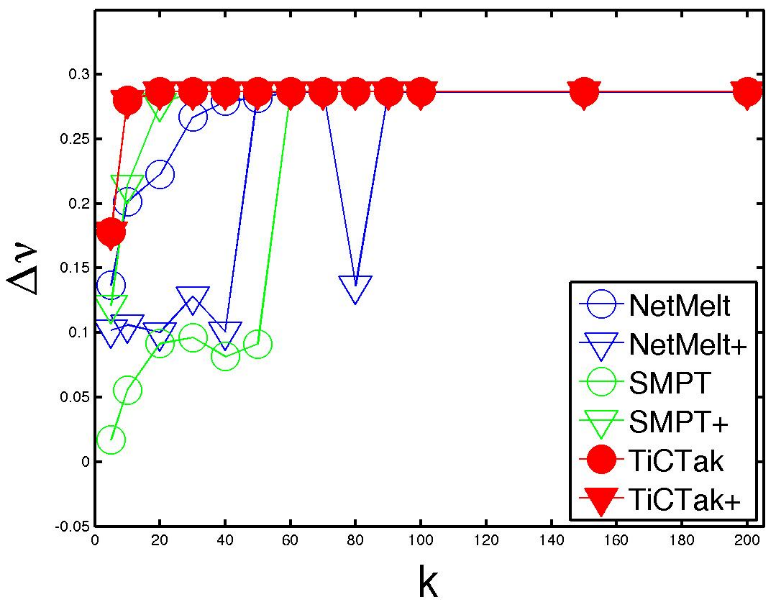
(4) Collaboration



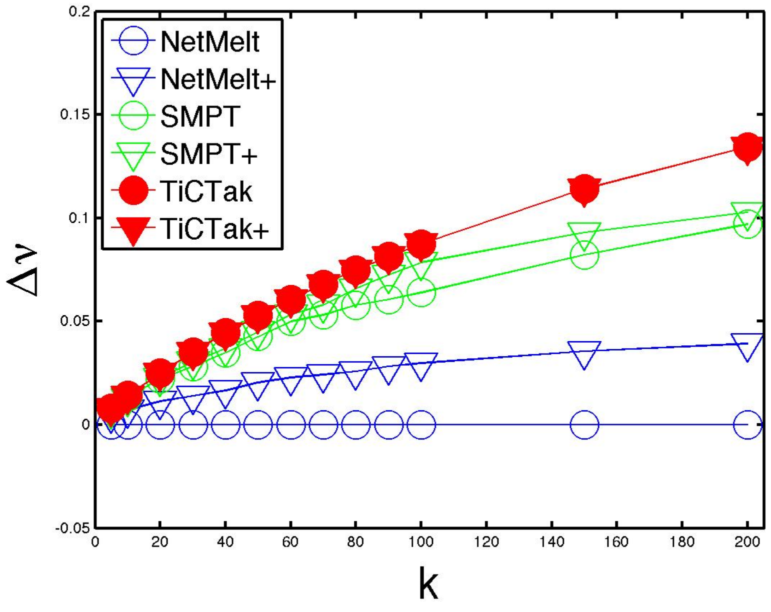
(5) Airport

Note: In Figure 4(4), *SMPT*, *SMPT+*, *NetMelt* overlapping with the *TiCTak* and *TiCTak+*.

Figure 5: Effectiveness of TiCTak, TiCTak+ and Other Algorithms for All Experimental Datasets

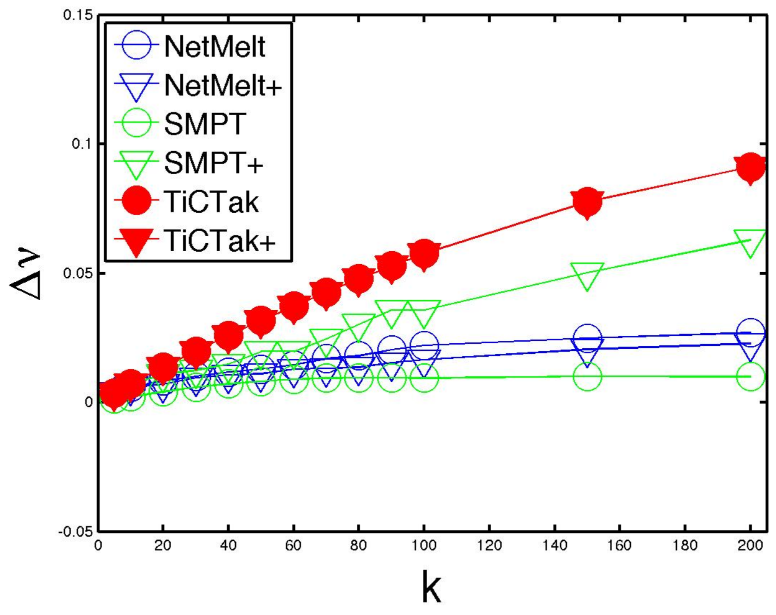


(6) Power

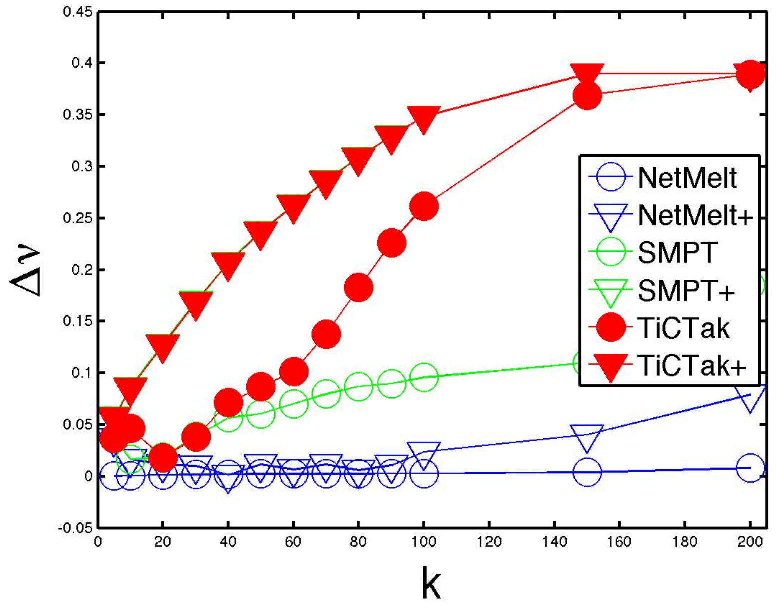


(7) Email

Figure 5: Effectiveness of TiCTak, TiCTak+ and Other Algorithms for All Experimental Datasets



(8) LiveJournal



(9) Router

Figure 5: Effectiveness of TiCTak, TiCTak+ and Other Algorithms for All Experimental Datasets

Note: In Figure 4(9), $SMPT+$ overlaps with the $TiCTak+$.

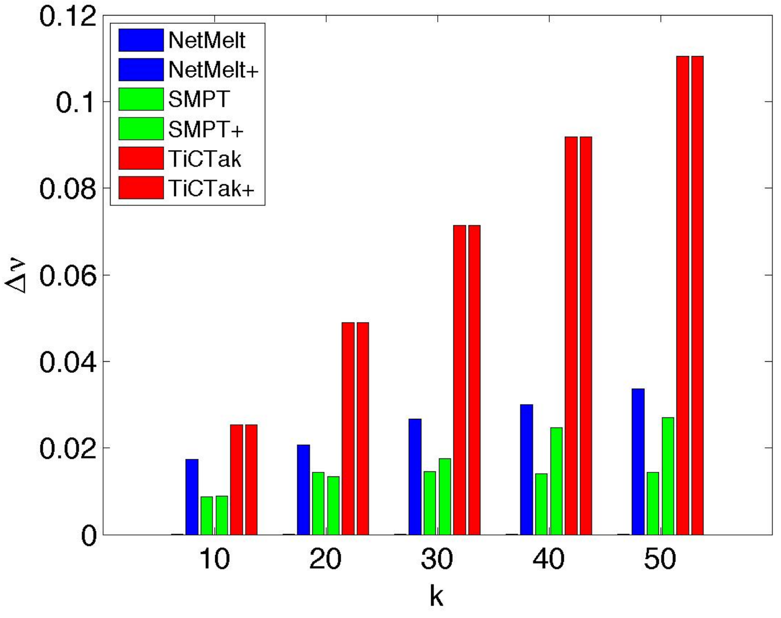


Figure 6: Effectiveness of TiCTak, TiCTak+ and Other Algorithms for Facebook Dataset

Figure 6 zooms into part of experiment results of Facebook dataset in Figure 4(1) using a bar chart. It provides a clearer visual comparison when budget is less than or equal to 50 edges. It is observed that $TiCTak$ and $TiCTak$+ curves are steeper than other methods and result in more and more performance gain as budget k increases. The incremental rate of $TiCTak$ and $TiCTak$+ is around 0.002 per edge where the other methods are less than 0.0001 per edge.

4.3 Efficiency of TiCTak and TiCTak+ Algorithms

In this section, the efficiency of the proposed $TiCTak$ and $TiCTak$+ algorithms along with $SMPT$, $SMPT$+, $NetMelt$ and $NetMelt$+ algorithms are evaluated. To measure the efficiency, the computation time and the decrease of eigen-centrality simultaneously for a given budget are evaluated.

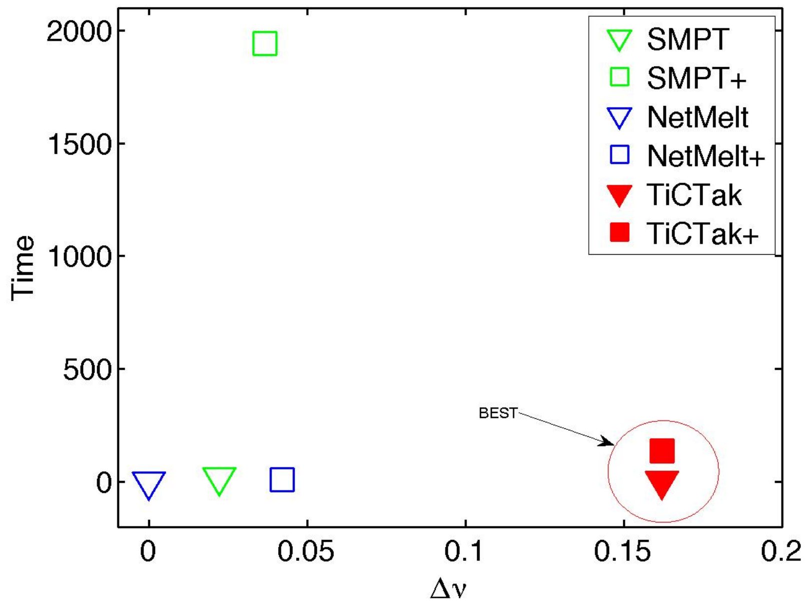
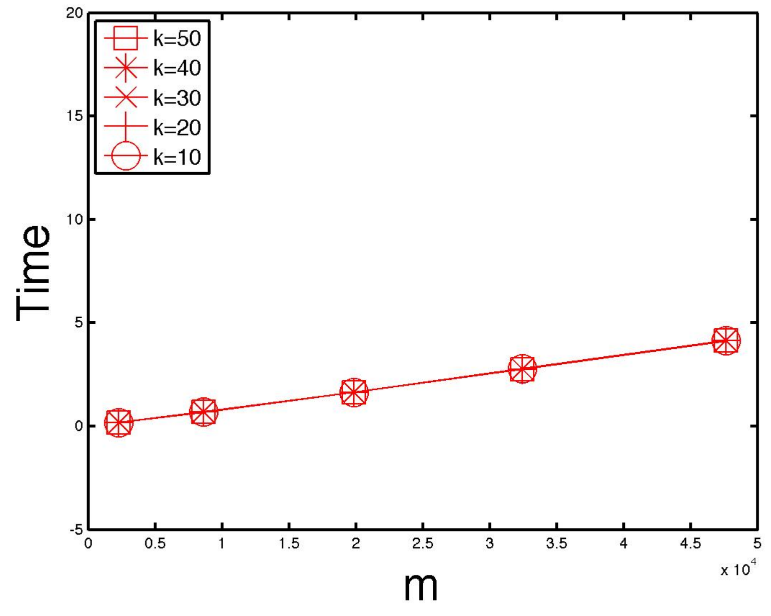
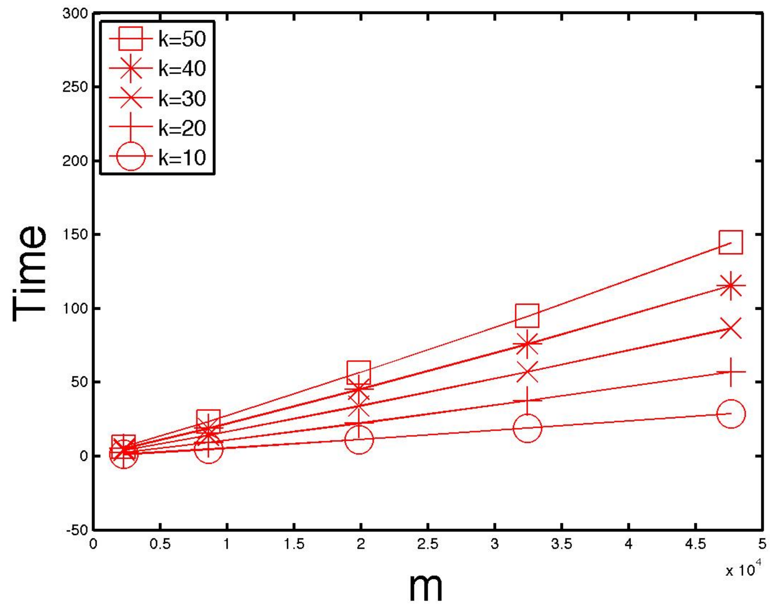


Figure 7: Efficiency of TiCTak, TiCTak+ and Other Algorithms for Facebook Dataset

The efficiency for the six algorithms are evaluated. Figure 7 shows the algorithm efficiency where the x-axis represents the decrease of the eigen-centrality and the y-axis represents the computation time when budget is 100. $SMPT$, $NetMelt$ and $NetMelt$+ have low computation time but less eigen-centrality reduction. $SMPT$+ has high computation time and less eigen-centrality reduction. The points $TiCTak$ and $TiCTak$+ are in the lower right corner of the chart. The $TiCTak$ and $TiCTak$+ use the least time and lead to the biggest decrease of the eigen-centrality. Therefore, using the proposed algorithm, there is no need to trade off between solution quality and computation time because the proposed algorithm achieve the best decrease of eigen-centrality and computation time at the same time.



(1) TiCTak



(2) TiCTak+

Figure 8: Scalability of TiCTak and TiCTak+ Algorithms

The subsets of the largest data set LiveJournal (m = 2307, 8621, 19854, 32443, 47641) is used to evaluate the scalability of the proposed algorithms. The results are presented in Figure 8. It can be seen that the proposed $TiCTak$ and $TiCTak$+ scale almost near linearly with respect to m, which means that they are suitable for large graphs.

Chapter 5

SOFTWARE PACKAGES

In this thesis, the software packages GCO_Melt_Nodes, GCO_Melt_Edges and GCO_TiCTak_Edges are used for the simulation of different algorithms while the second and third packages are developed as part of the work of this research.

5.1 GCO_Melt_Nodes

As shown in Table 5, GCO_Melt_Nodes package supports ten methods to select and delete k-vital nodes on large graphs.

The input includes an adjacency matrix $\mathbf{A}$, a budget k and a method name (shown in Table 5). The outputs are a new adjacency matrix $\mathbf{A}$ and a set $\mathcal{S}$ including k nodes. Here is the formulation:

$$[\mathbf{A}, \mathcal{S}] = GCO\_Melt\_Nodes(\mathbf{A}, k,' Method')$$

'NetShield' is picking the nodes with the highest 'NetShield' scores (Tong et al. 2010). 'Degrees' is picking the nodes with the highest degrees. 'ShortestPath' is picking the nodes with the highest betweenness centrality scores based on the shortest path (Freeman 1977). 'NewmanRandomWalk' is picking the nodes with the highest betweenness centrality scores based on random walks (Newman 2005). 'PageRank' is picking the nodes with the highest PageRank (Page et al. 1999). 'LeadingEigenvalue' is picking the nodes with the highest eigen scores (Chakrabarti et al. 2008). '2ndEigen-

Table 5: 'Method' of GCO_Melt_Nodes

'Method'	Method
NETSHIEL	NetShield
NETSHIRE	NetShieldRestart
SUMMATIO	Degrees
SHORTPAT	ShortestPath
NEWMANRW	NewmanRandomWalk
NEWMRWRE	NewmanRandomWalkRestart
PAGERANK	PageRank
FRISTEIG	LeadingEigenvalue
SECONDVE	2ndEigenvector
ABNORMAL	Abnormality

vector' is picking the nodes with the second highest eigenvector scores. 'Abnormality' is picking the nodes with the highest abnormality scores (Sun et al. 2005).

5.2 GCO_Melt_Edges

As shown in Table 6, *GCO_Melt_Edges* package supports 26 methods to select and delete k-vital edges on large graphs.

The inputs include an adjacency matrix $\mathbf{A}$, a budget k and a method name (show in Table 6). The outputs are a new adjacency matrix $\mathbf{A}$ and a set $\mathcal{S}$ including k edges. Here is the formulation:

$$[\mathbf{A}, \mathcal{S}] = GCO\_Melt\_Edges(\mathbf{A}, k,' Method')$$

In Table 6, *SMP* algorithm is standard matrix-perturbation based algorithm to update eigenpairs. As mentioned before, *UdEigen* is a algorithm to update eigen-centrality. *UdEigen* can also be used to update eigenvalues because Step 4 of *UdEigen* can update the eigenvalues. Eigen-centrality is a evaluation of the importance of node(Newman 2008). Leading eigenvalue (Wang et al. 2003; Prakash et al. 2012),

Table 6: 'Method' of GCO_Melt_Edges

'Method'	Update Eigen-centrality	Update Eigenvalues	Evaluation	Restart
CTOFIRST	-	$UdEigen$	Leading Eigenvalue	No
CTRFIRST	-	$UdEigen$	Leading Eigenvalue	Yes
CTOROBUS	-	$UdEigen$	Robustness	No
CTRROBUS	-	$UdEigen$	Robustness	Yes
CTOTRIAN	-	$UdEigen$	Triangles	No
CTRTRIAN	-	$UdEigen$	Triangles	Yes
CTOVECTO	$UdEigen$	-	Leading Eigen-centrality	No
CTRVECTO	$UdEigen$	-	Leading Eigen-centrality	Yes
CTOVEC10	$UdEigen$	-	Top 10^{th} Eigen-centrality	No
CTRVEC10	$UdEigen$	-	Top 10^{th} Eigen-centrality	Yes
CTOVE10S	$UdEigen$	-	Top 10 Eigen-centrality	No
CTRVE10S	$UdEigen$	-	Top 10 Eigen-centrality	Yes
MBOFIRST	-	SMP	Leading Eigenvalue	No
MBRFIRST	-	SMP	Leading Eigenvalue	Yes
MBOROBUS	-	SMP	Robustness	No
MBRROBUS	-	SMP	Robustness	Yes
MBOTRIAN	-	SMP	Triangles	No
MBRTRIAN	-	SMP	Triangles	Yes
MBOVECTO	SMP	-	Leading Eigen-centrality	No
MBRVECTO	SMP	-	Leading Eigen-centrality	Yes
MBOVEC10	SMP	-	Top 10^{th} Eigen-centrality	No
MBRVEC10	SMP	-	Top 10^{th} Eigen-centrality	Yes
MBOVE10S	SMP	-	Top 10 Eigen-centrality	No
MBRVE10S	SMP	-	Top 10 Eigen-centrality	Yes
NETMELTO	-	-	$NetMelt$	No
NETMELTR	-	-	$NetMelt$	Yes

robustness(Jun et al. 2010) and triangles(Tsourakakis 2008, 2011) are evaluations of connective of edges.

'NETMELTO' is picking the edges with the highest 'NetMelt' scores (Tong et al. 2012). 'NETMELTR' is 'NETMELTO' with restart. 'CTOFIRST' uses $UdEigen$ to update eigenvalues and picks the edges leading to the biggest decrease of leading eigenvalue. 'CTRFIRST' is 'CTOFIRST' with restart. 'CTOROBUS' uses $UdEigen$

to update eigenvalues and picks the edges leading to the biggest decrease of robustness. 'CTRROBUS' is 'CTOROBUS' with restart. 'CTOTRIAN' uses $UdEigen$ to update eigenvalues and picks the edges leading to the biggest decrease of triangles. 'CTRTRIAN' is 'CTOTRIAN' with restart. 'CTOVECTO' uses $UdEigen$ to update eigen-centrality and picks the edges leading to the biggest decrease of leading eigen-centrality. 'CTRVECTO' is 'CTOVECTO' with restart. 'CTOVEC10' uses $UdEigen$ to update eigen-centrality and picks the edges leading to the biggest decrease of top 10^{th} eigen-centrality. 'CTRVEC10' is 'CTOVEC10' with restart. 'CTOVEC10S' uses $UdEigen$ to update eigen-centrality and picks the edges leading to the biggest decrease of top 10 eigen-centrality. 'CTRVEC10' is 'CTOVEC10S' with restart. 'MBOFIRST' uses SMP to update eigenvalues and picks the edges leading to the biggest decrease of leading eigenvalue. 'MBRFIRST' is 'MBOFIRST' with restart. 'MBOROBUS' uses SMP to update eigenvalues and picks the edges leading to the biggest decrease of robustness. 'MBRROBUS' is 'MBOROBUS' with restart. 'MBOTRIAN' uses SMP to update eigenvalues and picks the edges leading to the biggest decrease of triangles. 'MBRTRIAN' is 'MBOTRIAN' with restart. 'MBOVECTO' uses SMP to update eigen-centrality and picks the edges leading to the biggest decrease of leading eigen-centrality. 'MBRVECTO' is 'MBOVECTO' with restart. 'MBOVEC10' uses SMP to update eigen-centrality and picks the edges leading to the biggest decrease of top 10^{th} eigen-centrality. 'MBRVEC10' is 'MBOVEC10' with restart. 'MBOVEC10S' uses SMP to update eigen-centrality and picks the edges leading to the biggest decrease of top 10 eigen-centrality. 'MBRVEC10' is 'MBOVEC10S' with restart.

5.3 GCO_TiCTak_Edges

As shown in Table 7, *GCO_TiCTak_Edges* package supports four methods to select and delete k edges leading to the biggest decrease of eigen-centrality of target nodes.

The inputs are an adjacency matrix $\mathbf{A}$, a budget k, the target nodes set $\mathcal{I}$ and a method name (shown in Table 7). The outputs are a new adjacency matrix $\mathbf{A}$ and a set $\mathcal{S}$ including k edges. Here is the formulation:

$$[\mathbf{A}, \mathcal{S}] = GCO\_TiCTak\_Edges(\mathbf{A}, k, \mathcal{I}, 'Method')$$

Table 7: 'Method' of GCO_TiCTak_Edges

'Method'	Method
TICTAK	TiCTak
TICTRE	TiCTak+
SMPTOR	SMPT
SMPTRE	SMPT+

'TiCTak' is *UdEigen* based method to pick the edges leading to the biggest decrease of eigen-centrality of target nodes. 'TiCTak' is 'TiCTak' with restart. 'SMPT' is *SMP* based method to pick the edges leading to the biggest decrease of eigen-centrality of target nodes. 'SMPT+' is 'SMPT' with restart.

Chapter 6

RELATED WORK

This chapter reviews the related work, which can be categorized in three parts, (a) centrality measures, (b) connectivity optimization, and (c) graph mining.

There are a number of centrality measures, such as, eigen-centrality (Newman 2008), shortest path based centrality (Freeman 1977), PageRank (Page et al. 1999), HITS (Kleinberg 1999), shield value (Tong et al. 2010), and random walks based centrality (Newman 2005; Kang et al. 2011), etc. In this thesis, I use eigen-centrality.

Connectivity optimization is a hot topic in recent years. In (Tong et al. 2010), Tong et. al. proposed an effective node immunization algorithm for the SIS model by approximately minimizing the leading eigenvalue. In (Valler et al. 2011; Prakash et al. 2010), Prakash et. al. proposed effective algorithms to perform node immunization on time-varying graphs. In (Tong et al. 2012), Tong et. al. proposed effective algorithms to optimize the leading eigenvalue that controls the information dissemination process. Other algorithms for controlling the information dissemination include the influence maximization (Kempe, Kleinberg, and Tardos 2003; Datta, Majumder, and Shrivastava 2010; Chen, Wang, and Wang 2010), and finding effectors in social networks (Lappas et al. 2010), etc. These algorithms are not target-specific while proposed $TiCTak$ and $TiCTak+$ are target-specific, which can contain the dissemination of entities from a set of target nodes.

Representative graph mining works include frequent substructure discovery (Xin et al. 2005; Jin et al. 2005), community mining and graph partition (Karypis and Kumar 2000; Backstrom et al. 2006), proximity (Tong, Faloutsos, and Pan 2006;

Geerts, Mannila, and Terzi 2004; Tong et al. 2006), the bridge centrality (Hwang et al. 2008), bridgeness based detection of fuzzy communities (Nepusz et al. 2008), the network value of a customer (Domingos and Richardson 2001), graph blocker (Habiba and Berger-Wolf 2008), the connectivity of the small world (Shi et al. 2008) and social capital (Licamele and Getoor 2006), etc.

Chapter 7

CONCLUSION

In this thesis, I study the problem of target-specific centrality manipulation on large networks.

After introduce the background of the research, the intervened aspect of node centrality is defined, i.e., to minimize the eigen-centrality of target nodes by deleting edges.

I reviewed two algorithms to update eigenpairs. The first algorithm is standard matrix-perturbation theory, which can approximately update eigenpairs. The second algorithm, called *Cheetah*, can accurately update eigenpairs. However, the time complexity of *Cheetah* is $O(nt^2)$, which prohibits its use for large networks.

Therefore, I improved *Cheetah* and proposed an algorithm, called *UdEigen*, to efficiently update eigen-centrality. The time complexity of *UdEigen* is $O(t^3)$, better than *Cheetah*.

In addition, accurate and efficient algorithms *TiCTak* and *TiCTak+* are proposed to select and delete edges to minimize the eigen-centrality, with a linear time complexity. The time complexity of *TiCTak* is $O(mt^3)$ and that of *TiCTak* is $O(kmt^3)$.

Extensive evaluations are conducted on a diverse set of real networks, which consistently demonstrate the effectiveness (shown in Figure 5,6), efficiency (shown in Figure 7) and scalability (shown in Figure 8) of the proposed algorithm. For effectiveness, *TiCTak* and *TiCTak+* are consistently outperform the rest of algorithms. Both algorithms are 261% on average more effective in terms of eigen-centrality than the best baseline. For efficiency, *TiCTak* and *TiCTak+* use the least time and lead to the biggest decrease of the eigen-centrality. There is no need to trade off between solution quality and computation time because the proposed algorithm achieve the best decrease of eigen-centrality and computation time at the same time. For scalability, $TiCTak$ and $TiCTak$+ scale almost near linearly with respect to m, which means that they are suitable for large graphs.

$TiCTak$ and $TiCTak$+ have equally good performance except in the Router dataset, shown in Figure 5(9), where $TiCTak$+ leads to the biggest decrease of the eigen-centrality. Why does the experimental results of $TiCTak$+ are better than that of $TiCTak$ in this kind of dataset? In the future, I will do more experiments in different datasets and analyze datasets' structure to find out the solution.

REFERENCES

Backstrom, Lars, Dan Huttenlocher, Jon Kleinberg, and Xiangyang Lan. 2006. “Group formation in large social networks: membership, growth, and evolution.” In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining,* 44–54. ACM.

Chakrabarti, Deepayan, Yang Wang, Chenxi Wang, Jurij Leskovec, and Christos Faloutsos. 2008. “Epidemic thresholds in real networks.” *ACM Transactions on Information and System Security (TISSEC)* 10 (4): 1.

Chan, Hau, Leman Akoglu, and Hanghang Tong. 2014. “Make It or Break It: Manipulating Robustness in Large Networks.” In *SDM,* 325–333. SIAM.

Chen, Chen, and Hanghang Tong. 2016. “On the eigen-functions of dynamic graphs: Fast tracking and attribution algorithms.” *Statistical Analysis and Data Mining: The ASA Data Science Journal.*

Chen, Wei, Chi Wang, and Yajun Wang. 2010. “Scalable influence maximization for prevalent viral marketing in large-scale social networks.” In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining,* 1029–1038. ACM.

Datta, Samik, Anirban Majumder, and Nisheeth Shrivastava. 2010. “Viral marketing for multiple products.” In *2010 IEEE International Conference on Data Mining,* 118–127. IEEE.

Domingos, Pedro, and Matt Richardson. 2001. “Mining the network value of customers.” In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining,* 57–66. ACM.

Freeman, Linton C. 1977. “A set of measures of centrality based on betweenness.” *Sociometry:* 35–41.

Geerts, Floris, Heikki Mannila, and Evimaria Terzi. 2004. “Relational link-based ranking.” In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30,* 552–563. VLDB Endowment.

Habiba, H., and T. Y. Berger-Wolf. 2008. “Graph Theoretic Measures for Identifying Effective Blockers of Spreading Processes in Dynamic Networks.”

Hwang, Woochang, Taehyong Kim, Murali Ramanathan, and Aidong Zhang. 2008. “Bridging centrality: graph mining from element level to group level.” In *Proceed-*

ings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, 336–344. ACM.

Jin, Ruoming, Chao Wang, Dmitrii Polshakov, Srinivasan Parthasarathy, and Gagan Agrawal. 2005. "Discovering frequent topological structures from graph datasets." In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining,* 606–611. ACM.

Jun, Wu, Mauricio Barahona, Tan Yue-Jin, and Deng Hong-Zhong. 2010. "Natural connectivity of complex networks." *Chinese physics letters* 27 (7): 078902.

Kang, U, Spiros Papadimitriou, Jimeng Sun, and Hanghang Tong. 2011. "Centralities in Large Networks: Algorithms and Observations." In *SDM,* 2011:119–130. SIAM.

Karypis, George, and Vipin Kumar. 2000. "Multilevel k-way hypergraph partitioning." *VLSI design* 11 (3): 285–300.

Kempe, David, Jon Kleinberg, and Éva Tardos. 2003. "Maximizing the spread of influence through a social network." In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining,* 137–146. ACM.

Kleinberg, Jon M. 1999. "Authoritative sources in a hyperlinked environment." *Journal of the ACM (JACM)* 46 (5): 604–632.

Lappas, Theodoros, Evimaria Terzi, Dimitrios Gunopulos, and Heikki Mannila. 2010. "Finding effectors in social networks." In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining,* 1059–1068. ACM.

Li, Liangyue, Hanghang Tong, Yanghua Xiao, and Wei Fan. 2015. "Cheetah: fast graph kernel tracking on dynamic graphs." In *SDM.* SIAM.

Licamele, Louis, and Lise Getoor. 2006. "Social capital in friendship-event networks." In *Sixth International Conference on Data Mining (ICDM'06),* 959–964. IEEE.

Nepusz, Tamás, Andrea Petróczi, László Négyessy, and Fülöp Bazsó. 2008. "Fuzzy communities and the concept of bridgeness in complex networks." *Physical Review E* 77 (1): 016107.

Newman, Mark EJ. 2005. "A measure of betweenness centrality based on random walks." *Social networks* 27 (1): 39–54.

———. 2008. "The mathematics of networks." *The new palgrave encyclopedia of economics* 2 (2008): 1–12.

Page, Lawrence, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. "The PageRank citation ranking: bringing order to the web."

Prakash, B Aditya, Deepayan Chakrabarti, Nicholas C Valler, Michalis Faloutsos, and Christos Faloutsos. 2012. "Threshold conditions for arbitrary cascade models on arbitrary networks." *Knowledge and information systems* 33 (3): 549–575.

Prakash, B Aditya, Hanghang Tong, Nicholas Valler, Michalis Faloutsos, and Christos Faloutsos. 2010. "Virus propagation on time-varying networks: Theory and immunization algorithms." In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases,* 99–114. Springer.

Shi, Xiaolin, Matthew Bonner, Lada A Adamic, and Anna C Gilbert. 2008. "The very small world of the well-connected." In *Proceedings of the nineteenth ACM conference on Hypertext and hypermedia,* 61–70. ACM.

Sun, Jimeng, Huiming Qu, Deepayan Chakrabarti, and Christos Faloutsos. 2005. "Neighborhood formation and anomaly detection in bipartite graphs." In *Fifth IEEE International Conference on Data Mining (ICDM'05),* 8–pp. IEEE.

Tong, Hanghang, Christos Faloutsos, and Jia-Yu Pan. 2006. "Fast random walk with restart and its applications."

Tong, Hanghang, Jingrui He, Mingjing Li, Wei-Ying Ma, Hong-Jiang Zhang, and Changshui Zhang. 2006. "Manifold-ranking-based keyword propagation for image retrieval." *EURASIP Journal on Advances in Signal Processing* 2006 (1): 1–10.

Tong, Hanghang, B Aditya Prakash, Tina Eliassi-Rad, Michalis Faloutsos, and Christos Faloutsos. 2012. "Gelling, and melting, large graphs by edge manipulation." In *Proceedings of the 21st ACM international conference on Information and knowledge management,* 245–254. ACM.

Tong, Hanghang, B Aditya Prakash, Charalampos Tsourakakis, Tina Eliassi-Rad, Christos Faloutsos, and Duen Horng Chau. 2010. "On the vulnerability of large graphs." In *2010 IEEE International Conference on Data Mining,* 1091–1096. IEEE.

Tsourakakis, Charalampos E. 2008. "Fast counting of triangles in large real networks without counting: Algorithms and laws." In *2008 Eighth IEEE International Conference on Data Mining,* 608–617. IEEE.

———. 2011. "Counting triangles in real-world networks using projections." *Knowledge and Information Systems* 26 (3): 501–520.

Valler, Nicholas C, B Aditya Prakash, Hanghang Tong, Michalis Faloutsos, and Christos Faloutsos. 2011. “Epidemic spread in mobile ad hoc networks: Determining the tipping point.” In *International Conference on Research in Networking,* 266–280. Springer.

Wang, Yang, Deepayan Chakrabarti, Chenxi Wang, and Christos Faloutsos. 2003. “Epidemic spreading in real networks: An eigenvalue viewpoint.” In *Reliable Distributed Systems, 2003. Proceedings. 22nd International Symposium on,* 25–34. IEEE.

Xin, Dong, Jiawei Han, Xifeng Yan, and Hong Cheng. 2005. “Mining compressed frequent-pattern sets.” In *Proceedings of the 31st international conference on Very large data bases,* 709–720. VLDB Endowment.

APPENDIX A

PROOF OF FORMULATIONS

Proof of (3.10)

$$
\begin{aligned}
||\mathbf{q}_1|| &= \sqrt{(\Delta\mathbf{U}(:,1) - \mathbf{U}\mathbf{r}_1)'(\Delta\mathbf{U}(:,1) - \mathbf{U}\mathbf{r}_1)} \\
&= \sqrt{(\Delta\mathbf{U}(:,1)' - \mathbf{r}_1'\mathbf{U}')(\Delta\mathbf{U}(:,1) - \mathbf{U}\mathbf{r}_1)} \\
&= \sqrt{1 - \Delta\mathbf{U}(:,1)'\mathbf{U}\mathbf{r}_1 - \mathbf{r}_1'\mathbf{U}'\Delta\mathbf{U}(:,1) + \mathbf{r}_1'\mathbf{U}'\mathbf{U}\mathbf{r}_1} \\
&= \sqrt{1 - \mathbf{r}_1'\mathbf{r}_1 - \mathbf{r}_1'\mathbf{r}_1 + \mathbf{r}_1'\mathbf{r}_1} \\
&= \sqrt{1 - \mathbf{r}_1'\mathbf{r}_1} \\
&= \sqrt{1 - ||\mathbf{r}_1||^2}
\end{aligned}
$$

Proof of (3.11)

$$
\begin{aligned}
||\mathbf{q}_2|| &= \sqrt{(\mathbf{x}' + \alpha\mathbf{q}_1')(\mathbf{x} + \alpha\mathbf{q}_1)} \\
&= \sqrt{\mathbf{x}'\mathbf{x} + \alpha\mathbf{x}'\mathbf{q}_1 + \alpha\mathbf{q}_1'\mathbf{x} + \alpha^2\mathbf{q}_1'\mathbf{q}_1} \\
&= \sqrt{1 - \mathbf{r}_2'\mathbf{r}_2 - \alpha\mathbf{r}_2'\mathbf{r}_1 - \alpha\mathbf{r}_1'\mathbf{r}_2 + \alpha^2(1 - \mathbf{r}_1'\mathbf{r}_1)} \\
&= \sqrt{1 - \mathbf{r}_2'\mathbf{r}_2 - (\mathbf{r}_1'\mathbf{r}_2)^2(1 + \mathbf{r}_1'\mathbf{r}_1)} \\
&= \sqrt{1 - ||\mathbf{r}_2||^2 - (\mathbf{r}_1'\mathbf{r}_2)^2(1 + ||\mathbf{r}_1||^2)}
\end{aligned}
$$

where $\mathbf{x} = \Delta\mathbf{U}(:,2) - \mathbf{U}\mathbf{r}_2$ and $\alpha = \mathbf{r}_1'\mathbf{r}_2$.