K-STEP BETWEENNESS CENTRALITY

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MELDA KEVSER AKGÜN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
INDUSTRIAL ENGINEERING

AUGUST 2019

Approval of the thesis:

K-STEP BETWEENNESS CENTRALITY

submitted by **MELDA KEVSER AKGÜN** in partial fulfillment of the requirements for the degree of **Master of Science in Industrial Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Yasemin Serin
Head of Department, **Industrial Engineering** _____

Assist. Prof. Dr. Mustafa Kemal Tural
Supervisor, **Industrial Engineering, METU** _____

Examining Committee Members:

Prof. Dr. Meral Azizoğlu
Industrial Engineering, METU _____

Assist. Prof. Dr. Mustafa Kemal Tural
Industrial Engineering, METU _____

Assoc. Prof. Dr. İsmail S. Bakal
Industrial Engineering, METU _____

Assist. Prof. Dr. Özgen Karaer
Industrial Engineering, METU _____

Assoc. Prof. Dr. Babek Erdebilli
Industrial Engineering, Ankara Yıldırım Beyazıt University _____

Date:

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Melda Kevser Akgün

Signature :

ABSTRACT

K-STEP BETWEENNESS CENTRALITY

Akgün, Melda Kevser
M.S., Department of Industrial Engineering
Supervisor: Assist. Prof. Dr. Mustafa Kemal Tural

August 2019, 90 pages

The notions of betweenness centrality (BC) and its extension group betweenness centrality (GBC) are widely used in social network analyses. We introduce variants of them; namely, the k-step BC and k-step GBC. The k-step GBC of a group of vertices in a network is a measure of the likelihood that at least one group member will get the information communicated between a randomly chosen pair of vertices through a randomly chosen shortest path within the first k steps of the start of the communication. The k-step GBC of a single vertex is the k-step BC of that vertex. The introduced centrality measures may find uses in applications where it is important or critical to obtain the information within a fixed time of the start of the communication. For the introduced centrality measures, we propose an algorithm that can compute successively the k-step GBC of several groups of vertices. Moreover, we propose a mixed integer programming formulation to compute the group that has the highest k-step GBC value and a heuristic approach to compute a group of vertices whose k-step GBC value is high. The performances of the proposed algorithms are evaluated through computational experiments on real and randomly generated networks.

Keywords: Betweenness centrality, Group betweenness, Network analysis, Social networks, k-step betweenness

ÖZ

K-ADIM ARASINDALIK MERKEZİLİĞİ

Akgün, Melda Kevser
Yüksek Lisans, Endüstri Mühendisliği Bölümü
Tez Yöneticisi: Dr. Öğr. Üyesi. Mustafa Kemal Tural

Ağustos 2019 , 90 sayfa

Arasındalık merkeziliği ve onun bir türevi olan grup arasındalık merkeziliği sosyal ağların analizinde yaygın olarak kullanılmaktadır. Bu çalışmada, onların türevleri olan k-adım arasındalık merkeziliği ve k-adım grup arasındalık merkeziliği tanıtılmaktadır. Bir ağdaki bir grup düğümün k-adım grup arasındalık merkeziliği, rastgele seçilen bir çift düğüm arasında rastgele seçilen bir en kısa yol üzerinden iletilen bilginin, grup üyelerinden en az bir tanesine iletişim başladıktan sonra ilk k adımda ulaşması olasılığının ölçütüdür. Tek bir düğümün k-adım grup arasındalık merkeziliği, o düğümün k-adım arasındalık merkeziliğidir. Tanıtılan merkezilik ölçütü, iletişim başlangıcından itibaren belirli bir sürede bilgiye ulaşılmasının kritik ya da önemli olduğu durumlarda faydalı olabilir. Tanıtılan merkezilik ölçütleri için, alt kümelerin k-adım grup arasındalık merkeziliğini art arda hesaplayan bir algoritma öneriyoruz. Ayrıca k-adım grup arasındalık merkeziliği en yüksek grubu bulmak için bir tam sayılı programlama modeli ve k-adım grup arasındalık merkeziliği yüksek olan bir grup bulmak için bir sezgisel yaklaşım öneriyoruz. Önerilen algoritmaların performansları, gerçek ve rastgele üretilen ağlar üzerinde değerlendirilmiştir.

Anahtar Kelimeler: Arasındalık merkeziliği, Grup arasındalık merkeziliği, Ağ analizi, Sosyal ağlar, k-adım arasındalık merkeziliği

To my family..

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor Assist. Prof. Dr. Mustafa Kemal Tural for his endless support, and excellent guidance during my MS study. I am really grateful for his time and patience. I learned a lot from him about how to conduct research, and it was a big honor for me to work with him.

I would like to thank the examining committee members of my thesis Prof. Dr. Meral Azizoğlu, Assoc. Prof. Dr. İsmail S. Bakal, Assist. Prof. Dr. Özgen Karaer, and Assoc. Prof. Dr. Babek Erdebilli for their valuable feedback.

I wish to thank my parents for their support. I feel very lucky for being an MS student at Middle East Technical University, Industrial Engineering Department. It was a great experience for me.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

BC	Betweenness centrality
GBC	Group betweenness centrality
NB	Normalized betweenness centrality
NGB	Normalized group betweenness centrality
CC	Closeness centrality
DC	Degree centrality
EC	Eigenvector centrality
MIP	Mixed integer programming
LP	Linear programming
SSSP	Single source shortest path
SNA	Social network analysis

CHAPTER 1

INTRODUCTION

A social network consists of a set of vertices representing actors and a set of edges representing relations between them. The concept of centrality plays a crucial role in the analyses of social networks. Several centrality measures have been proposed in the literature to determine important actors in a social network. See [1] for a review of different centrality measures and their application areas.

Some of the centrality measures evaluate the importance of an actor as a function of the number of times s/he acts as a bridge along the shortest paths connecting different vertices of the network, e.g., the stress and betweenness centralities [1]. The betweenness centrality (BC), one of the most commonly used centrality measures, is formally defined for the first time by Freeman [2]. The BC of a vertex or an actor v, is defined as the total fraction of the shortest paths between vertex pairs that use or pass through v. The normalized version of the BC of an actor (see Chapter 2 for a formal definition) can be interpreted as the probability that the actor gets the information that is communicated between a pair of vertices chosen uniformly at random through one of the shortest paths connecting them. According to Freeman [3], an actor having a high betweenness centrality value has the potential to control the communication taking place through the network by withholding, distorting, or maintaining information in transition.

Until the study of Brandes [4], algorithms computing the BC of all vertices of a network had a running time of $O(n^3)$. Brandes introduces the notion of dependency which allows him to design an algorithm that can compute the same in $O(nm)$ time.

The notion of the BC of a vertex has also been generalized to measure the centrality

of groups of vertices. Given a group of vertices C, the co-betweenness centrality of C introduced in [5], measures the likelihood that the information that is communicated between two randomly chosen vertices through a shortest path will pass through all the vertices in C. The group betweenness centrality (GBC), introduced in [6], on the other hand measures the likelihood that the information will pass through at least one vertex in C.

The problem of finding a group of vertices of a given size g in a network that has the highest GBC is in general an NP-hard problem [7]. Brandes in [4] and Puzis et al. in [8] propose algorithms to compute the GBC of any given group of vertices. The latter algorithm can compute the GBC of a given group of vertices within reasonable times even for large scale networks. More details about these algorithms can be found in Chapter 4. Recently, Veremyev et al. [9] propose a mixed integer linear programming formulation for the problem of finding a group of g vertices in a network that has the highest GBC. This formulation, however, is not suited for large scale networks. Moreover, Puzis et al. [10] propose a $1 - 1/e$ approximation algorithm for the same problem.

In several applications, it is important or critical to obtain the information within a fixed time of the start of the communication in the network. For example, consider a criminal network that is going to be kept under surveillance via some listening devices that are to be placed at certain vertices of the network. Where should these devices be located to maximize the probability that the information that is going to be communicated between pairs of vertices of this network will be listened by the devices within the first k steps of the start of the communication? Similarly, in detecting and blocking the spread of malware in computer communication networks, rumors in social networks, or viruses in contact networks, it may be desirable to find the optimal locations of monitoring devices to get the information in transition (malware, rumors, or viruses) within a given time limit.

In this thesis, we introduce and study the k-step betweenness and group betweenness centralities which are variants of the classical betweenness and group betweenness centralities, respectively. The normalized version of the k-step BC of a vertex measures the probability that the vertex will get the information communicated between

a random pair of vertices through one of the shortest paths within the first k steps of the start of the communication. The k-step GBC of a group of vertices, on the other hand, measures how likely it is that at least one vertex in the group will get the information within the first k steps. Even though time constrained influence maximization problems have been studied in the literature, see e.g., [11, 12], no time constrained version of the BC has been proposed or studied to the knowledge of the authors.

Consider, for example, a computer, transportation, or a social network where shortest path routing is employed. The BC of a vertex in such a network measures the likelihood that the vertex will act as a bridge along the shortest path between two vertices in this network. In other words, BC quantifies the extent of the control the vertex has on communications in the network. The BC of a vertex, on the other hand, does not tell us when the vertex will get what is being communicated. The measure proposed in this study, namely the k-step BC, quantifies the likelihood that the vertex will get what is being communicated within at most k steps (hops) of the start of the communication. Put it differently, the k-step BC of a vertex is a measure assessing the possibility that the vertex will act as a bridge along the shortest paths controlling communication/propagation early after (within at most k hops) the start of communication/propagation. When the value of k is large enough, the classical BC of a vertex agrees with the k-step BC of the vertex. The k-step BC and its extension to a group of vertices, namely the k-step GBC, can be applied to any setting where the classical betweenness centrality can be applied and where it is important or critical to obtain what is being communicated in the network early after the start of the communication. As examples, the proposed measures can be applied in transportation networks for early warning of severe weather conditions (through road side equipment), in criminal networks for early detection/prevention of a terrorist attack, in computer networks for early control of information, viruses, or malware.

The organization of the next chapters is as follows. In Chapter 2, a literature review is presented, and the concepts of the BC and GBC are explained. In Chapter 3, the notation used throughout the thesis is given, and the k-step BC and k-step GBC are introduced. In Chapter 4, proposed algorithm to compute the k-step GBC value for a given group size g is provided. An illustrative example detailing the steps to compute the k-step group betweenness centrality of a single group is provided in Chapter

5. In Chapter 6, the results of the computational experiments performed on randomly generated and real-life networks are provided. In Chapter 7, alternative solution methods; namely, a mixed integer programming formulation and a heuristic approach are provided. Finally, conclusion and future research directions are given in Chapter 8.

CHAPTER 2

LITERATURE REVIEW

2.1 Social Network Analysis

A social network consists of a set of vertices representing actors and a set of edges representing relations between them. Actors can be people, organizations, teams etc., and the relationship could be any kind of interaction, such as friendship, co-working relation, liking, trust, and kinship.

Recently, with the increase of mobile applications and sites like Facebook, LinkedIn, and Instagram social networks have become an important part of daily life for lots of people. This increasing interest resulted in the curiosity to investigate, evaluate, and identify social networks. Social network analysis (SNA) is an approach to investigate and visualize network structure within a social context. SNA helps to understand connections and behavior of actors, such as who is connected to whom, and to what extent, how information moves, and which actors control the information flow [13–15].

SNA is firstly introduced to investigate sociology and anthropology, but later it is used to analyze networks in several fields. Levin and Cross [16] use SNA to investigate how the strength of the ties affects knowledge transmission. They observe that strong ties have a higher capability to transfer important knowledge, and weak ties tend to transfer non-redundant knowledge and act as brokers between strong ties. Meltzer et al. [17] use the SNA measures to define principles for constituting a clinical improvement team of high quality. Lie et al. [18] investigate the effect of the cooperation in a virtual learning group, and propose SNA as an approach to define problems in an interactive education system, and improve efficiency. Korkmaz and Singh [19] inves-

tigate designing an effective team in the architecture, engineering, and construction industry. They use SNA to evaluate team integrity. Di Marco et al. use SNA in [20] to investigate the role of the team member who acts as a bridge by comparing two design project teams in India. Both of the teams consist of Indian and Americans, but in one of them, there is an Indian member who lived in the United States.

The concept of centrality plays a crucial role in SNA. For example, according to Ruan et al. [21] position of the vertices are important representatives for networks, and centrality is useful to guess the importance of vertices. If a network has high centrality values, it doesn't have distributed configurations, and a small proportion of all vertices can control most of the flows in the network [22, 23]. In a social network which has high centrality, a great majority of the vertices are connected with central vertices [24]. Therefore, actors with high centrality value are more powerful to affect others.

Several centrality measures have been proposed in the literature to determine important actors and essential transactions in social networks. Now, we explain the concept of centrality and its application areas.

2.2 Centrality Concepts

The term "centrality" is introduced by Bavelas in 1948 [25]. He uses it to investigate communications in small groups and claims that structural centrality of a vertex is correlated with the influence of that vertex on the other vertices in the group. Leavitt at al. [26] report study of Bavelas in detail, and they reach the conclusion that high centrality means high ability to problem-solving, leadership, and high job satisfaction. Studies of Bavelas and Leavitt are pioneers of lots of experimental and theoretical studies on centrality metrics.

Cohn and Marriott [27] use the term centrality to investigate political integration in the diversity of Indian social life. They question how a multiracial country could be administered, and they conclude that network centers hold important positions to interconnect Indian society. Pitts [28] analyzes the role of centrality on paths development in medieval cities and reconstruction of the transportation system. Beauchamp [29]

discusses limitations of centrality measure proposed by Bavelas [25], and improves centrality index. He claims that the association of sub-units based on centrality could optimize the organization. The term has been used by Czepiel in [30] to investigate convection of the technological innovation in the steel industry. Rogers [31] studies on types of centrality in inter-organizational relations. Coles uses centrality in [32] as a social network analysis technique to investigate the criminal network. Bruun and Brewe [33] investigate the relationship between students' communication and performance by using centrality metric.

The term centrality is first proposed for social network analysis. However, it has been used for so many applications such as anthropology, biology, traffic, facility location problem, highway-node routing, web page ranking, or prediction of polls. Even though there are different comments about its definition and method of measurement, it is a general idea that centrality is an important measure to identify the structure of networks [3].

2.2.1 Some Variants of Centrality

There are various centrality definitions in the literature to determine the most important and effective vertex in the network based on distance, feedback, path count, and so on. Four main centrality measures which have been widely used are degree, closeness, eigenvector, and betweenness centrality.

For the following definitions consider a connected graph $G = (V, E)$ with $|V| = n$ vertices and $|E| = m$ edges, and $\{s, t, v\} \in V$.

- **Degree Centrality**:

  This centrality measure is firstly introduced by Shaw [34], and long after Nieminen [35] offers a formal definition. Freeman [2] formulates a mathematical model to define degree centrality which is based on the links connected to a vertex. Despite its simplicity, this centrality measure is still pretty useful in applications. It is used for facility location problems. For directed networks indegree or outdegree could be used according to the direction of the link between related vertices, but using the formulation by this way is rare in real-life

applications. The degree centrality of a vertex v, denoted by DC(v), is defined as

$$DC(v) = deg(v), \tag{2.1}$$

where $deg(v)$ is the number of vertices adjacent to v.

Calculation of degree centrality has $O(m)$ time complexity for all vertices [1].

- **Closeness Centrality**:

  The term closeness centrality is firstly used by Bavelas [25], and is precisely defined by Sabidussi [36]. The closeness centrality of a vertex is the reciprocal of the sum of the distances between the vertex and all other vertices. It is used for service facility location problems. It takes into consideration the indirect relationship between vertices, unlike the degree centrality. This centrality measure is firstly defined for connected and unweighted networks, but Newman [37] generalizes formulation for weighted networks. The closeness centrality of a vertex v, denoted by CC(v), is defined as

$$CC(v) = \frac{1}{\sum_{s \in V} d(s, v)}, \tag{2.2}$$

  where $d(s, v)$ is the distance between s and v.

  Calculation of closeness centrality has $O(nm)$ time complexity for all vertices for unweighted networks [4].

- **Eigenvector Centrality**:

  Eigenvector centrality is introduced by Bonacich in [38], he explains unique properties of the eigenvector centrality in a followup paper [39]. Eigenvector centrality takes into consideration properties of neighborhoods, unlike the degree centrality, so centrality of a vertex is evaluated according to the centralities of its adjacent vertices. So, connections with highly central vertices provide more contribution to the centrality of the vertex compared to connections with less central ones. It can be applied for disconnected networks by considering every connected component as a separate graph. It is also applicable

on weighted graphs, but there is no consensus about its interpretation. PageRank is the most familiar application of the eigenvector centrality. Google uses PageRank to rank website pages according to importance, it gives each page a score based on the count of received links from other pages. The eigenvector centrality of a vertex v, denoted by EC(v), is defined as

$$EC(v) = \frac{1}{\lambda} \sum_{s \in N(v)} EC(s), \tag{2.3}$$

where λ is a constant.

Calculation of eigenvector centrality has $O(n^2)$ time complexity for all vertices [1].

- **Betweenness Centrality**:

  One of the most featured centrality measures is betweenness centrality. In this study, we focus on betweenness centrality, group betweenness centrality, and a derivation that we introduce in this thesis, k-step BC and k-step GBC. Hence, we explain the terms BC and GBC in more detail below and we give formal definitions in Chapter 3.

2.2.2 Betweenness Centrality

Betweenness centrality, one of the most commonly used centrality measures, is formally defined for the first time by Freeman [2]. The BC of a vertex v, is defined as the total fraction of the shortest paths between vertex pairs that use or pass through v. The normalized version of the BC of a vertex can be interpreted as the probability that the vertex gets the information that is communicated between a pair of vertices chosen uniformly at random through one of the shortest paths connecting them. According to Freeman [3], an actor having a high betweenness centrality value has the potential to control the communication taking place through the network by withholding, distorting, or maintaining information in transition. A formal definition and details about the calculation of BC are presented in Chapter 3.

Yan et al. [40] use BC to define a new routing strategy to decrease traffic congestion. Also, Jayaweera et al. use BC in [41] to define locations which mostly affect

traffic congestion in Sri Lanka. Newman and Girvan use BC [42] to identify a new network clustering algorithm in social and biological networks. Liu et al. [43] use degree, closeness, and betweenness centrality measures and ranking methods to evaluate co-authorship of JCDL research community, and they show that performance of betweenness centrality is better than other centrality measures, and correlated with ranking methods. Liu et al. use BC as a statistical tool to construct a weighted network of research areas in [44]. Sparrow [45] uses degree, closeness, and betweenness centrality to investigate the relationships and security holes in criminal networks. Grunspan et al. [46] use BC to investigate the relationship between students and the effect of the relationships on the learning outcomes. Magaia et al. [47] investigate the usage of the BC to determine the appropriate vertex to forward messages in delay tolerant networks.

Different variants of BC have been proposed in the literature [48]. The bounded-distance BC proposed by Borgatti and Everett [49] only considers shortest paths whose lengths are less than or equal to some positive integer K with the idea that long paths are seldom used for communication of information. Distance-scaled BC is another variant of the BC where the fraction of the shortest paths between a pair of vertices passing through an actor is divided by the length of the shortest paths to decrease the contribution of long paths to the BC of the actor [49]. Stress BC counts all shortest paths controlled by the considered vertex [48, 50]. Proximal source/target BC is introduced based on the idea that next-to-last vertex in the shortest path could be considered as representative of total control over the connection through this path [49]. Other centrality measures related to and inspired by the BC include the game-theoretic BC [51], the random walk BC [52], α-weight BC [53], Canonical-path BC [54], and the evacuation BC [55]. Grassi et al. [56] test several variants of BC in identifying leaders in criminal networks.

Most of the studies on BC in the literature, including this one, consider static networks. Some recent studies consider the problem of updating the BC in dynamic networks, see e.g., [57–59].

Brandes [4] proposes an algorithm which has $O(nm)$ time complexity to compute BC values of all vertices in the graph. This exact algorithm necessitates to solve single-

source shortest path problem for each vertex in the graph and is computationally expensive for large scale networks. Therefore, some approximation algorithms are proposed in the literature which consider random sampling, see e.g., [60], [54], [61], adaptive sampling [62], and local techniques [63].

In this chapter, we present the mathematical definition of four classical centrality measures; namely, degree centrality, closeness centrality, eigenvector centrality, and betweenness centrality. Now, we investigate the optimum vertices for each of them on a graph $G = (V, E)$ of order 19 given in Figure 2.1.

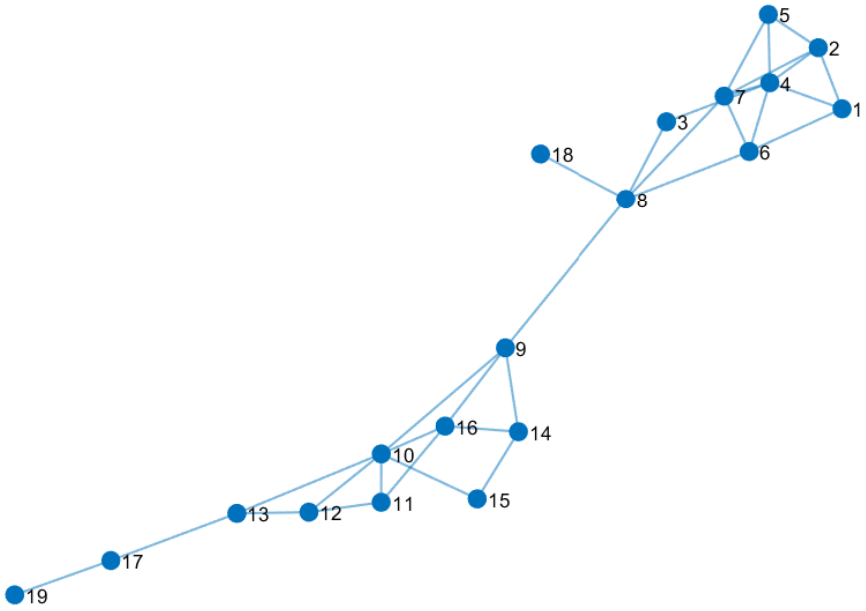


Figure 2.1: A graph of order $n = 19$

In G, the vertex with the highest betweenness centrality value is 8, the vertex with the highest closeness centrality value is 9, the vertex with the highest eigenvector centrality is 4, and the vertices with the highest degree centrality are 4 and 10. Consider vertices 4 and 10. Even though they have the same degree centrality values, vertex 4 has higher eigenvector centrality. This example clearly shows that different centrality measures could identify different vertices as the most central one.

2.2.3 Group Betweenness Centrality

The notion of the BC of a vertex has also been generalized to measure the centrality of groups of vertices. Given a group of vertices C, the co-betweenness centrality of C introduced by Kolaczky et al. [5], measures the likelihood that the information that is communicated between two randomly chosen vertices through a shortest path will pass through all the vertices in C. The group betweenness centrality (GBC), introduced by Everett and Borgatti in [6], on the other hand, measures the likelihood that the information will pass through at least one vertex in C.

In some studies, additional cohesiveness properties are imposed on the group members. For example, finding the most central (in terms of the GBC) group of vertices that form a κ-club is considered in [9], where a group of vertices form a κ-club if the distance between any two members of the group is less than or equal to κ. Similarly, [9] and [64] study the problem of finding a clique with the highest GBC, where a group of vertices form a clique if they form a 1-club.

GBC has been used in several applications. Ni et al. [65] use the GBC to investigate the role of disciplines in journal co-citation networks. Kchiche and Kamoun [66] use GBC to increase the performance of deployment strategies of roadside units in vehicular ad hoc networks. Puzis et al. [67] investigate the problem of finding the optimal locations for traffic monitoring units based on GBC. Tubi et al. [68] propose a new technique to prevent or slow down the propagation of computer worms and viruses in social networks by using the GBC. Moreover, Puzis et al. [69] propose a decision support tool that identifies the locations of malware filtering devices using the GBC to detect and block threat in large-scale communication networks. Guan et al. [70] propose a GBC-based cache location selection algorithm in a content-centric network to maximize cache delivery performance. One of the performance measures used is related to the number of hops (i.e., number of steps) it takes a content to reach its destination.

The problem of finding a group of vertices of a given size g in a network that has the highest GBC is in general an NP-hard problem [7]. Brandes [4] and Puzis et al. [8] propose algorithms to compute the GBC of any given group of vertices. The latter

algorithm can compute the GBC of a given group of vertices within reasonable times even for large scale networks. Recently, Veremyev et al. [9] propose a mixed integer linear programming formulation for the problem of finding a group of g vertices in a network that has the highest GBC. This formulation, however, is not suited for large scale networks. More details about these algorithms can be found in Chapter 4 Moreover, Dolev et al. [10] propose a $1 - 1/e$ approximation algorithm for the same problem.

CHAPTER 3

NOTATION AND PROBLEM DESCRIPTION

Let $G = (V, E)$ be a simple, unweighted, undirected, and connected graph with $|V| = n$ vertices and $|E| = m$ edges. Two vertices s and t in V are said to be adjacent if $\{s, t\} \in E$. A sequence of distinct vertices $s_0, s_1, \ldots, s_\ell$ is said to be a path of length ℓ joining s_0 to s_ℓ if the successive vertices in the sequence are adjacent. For any two vertices s and t in V, the distance between them is denoted by $d(s, t)$ and is defined as the length of a shortest path joining s to t. We denote by σ_{st} the number of shortest paths joining s to t. $d(s, s)$ and σ_{ss} are defined to be 0 and 1, respectively, for any $s \in V$. Moreover, we have that $d(s, t) = d(t, s)$ and $\sigma_{st} = \sigma_{ts}$ for any two vertices s and t in V. We denote by d and σ the distance and sigma matrices defined as $d = [d(s, t)]_{s,t \in V}$ and $\sigma = [\sigma_{st}]_{s,t \in V}$. Note that both of these matrices are symmetric and can be computed in $O(nm)$ time [4].

For any graph $G = (V, E)$, $md(G)$ represents the maximum possible distance in G, i.e., $md(G) = max_{s,t \in V} d(s, t)$. For any three vertices $s, t, v \in V$, we denote by $\sigma_{st}(v)$ the number of shortest paths joining s to t that pass through v. $\sigma_{st}(v)$ can be calculated as in Equation 3.1.

$$\sigma_{st}(v) = \begin{cases} \sigma_{sv}\sigma_{vt}, & \text{if } d(s, t) = d(s, v) + d(v, t) \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

For any three vertices $s, t, v \in V$ and any positive integer k, we denote by $\sigma^k_{st}(v)$ the number of shortest paths joining s to t that pass through v within the first k steps of

the path. More formally we have that

$$\sigma_{st}^{k}(v) = \begin{cases} \sigma_{sv}\sigma_{vt}, & \text{if } d(s,t) = d(s,v) + d(v,t) \,\&\, d(s,v) \leq k \\ 0, & \text{otherwise.} \end{cases} \tag{3.2}$$

We denote by $\delta_s(v)$ the dependency of vertex s on vertex v which is defined as

$$\delta_s(v) = \sum_{t \in V | t \neq s} \frac{\sigma_{st}(v)}{\sigma_{st}}. \tag{3.3}$$

This notion is introduced by Brandes [4]. In a similar manner, we introduce the k-step dependency of s on v as

$$\delta_s^k(v) = \sum_{t \in V | t \neq s} \frac{\sigma_{st}^k(v)}{\sigma_{st}}. \tag{3.4}$$

Note that if $d(s,v)$ is less than or equal to k, then it holds that $\delta_s^k(v) = \delta_s(v)$. Otherwise, we have that $\delta_s^k(v) = 0$. We denote by δ and δ^k the dependency and k-step dependency matrices defined as $\delta = [\delta_s(v)]_{s,v \in V}$ and $\delta^k = [\delta_s^k(v)]_{s,v \in V}$. Note in this case that these matrices need not be symmetric. Brandes [4] shows that $\delta_s(v)$ values can be computed recursively in $O(nm)$ time for all $s, v \in V$. Therefore the k-step dependency matrix δ^k can also be computed in $O(nm)$ time.

Now we explain betweenness and group betweenness centralities. We then introduce the k-step BC. Given a graph $G = (V, E)$ and a vertex $v \in V$, the BC of v represents the total control v has on pairwise communications between every pair of distinct vertices. It is assumed that the communication between a pair of vertices is realized through one of the shortest paths connecting them (selected uniformly at random). More formally, the BC of a vertex v, denoted by $B(v)$, is defined as

$$B(v) = \sum_{s,t \in V | s \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}. \tag{3.5}$$

A direct utilization of Equation 3.5 gives us an $O(n^3)$ time algorithm for the compu-

tation of $B(v)$ for every $v \in V$. Using the dependencies, we can write $B(v)$ as

$$B(v) = \sum_{s \in V} \delta_s(v). \tag{3.6}$$

As shown by Brandes [4], computing the dependency matrix δ in $O(nm)$ time followed by the computation of all BCs according to Equation 3.6 in $O(n^2)$ time results in an $O(nm)$ time algorithm to compute all BC values.

Betweenness centrality also has a probabilistic interpretation. $B(v)/\left(2\binom{|V|}{2}\right)$ represents the probability that the information sent from a vertex s chosen uniformly at random from V to a different vertex t chosen uniformly at random from $V \setminus \{s\}$ passes through v assuming that the information flows through one of the shortest paths joining s to t selected uniformly at random. We denote this probability by $NB(v)$ and call it the normalized BC of v. In summary, $NB(v)$ represents the probability that v gets the information sent through a randomly chosen shortest path between randomly chosen vertices.

Betweenness centrality can be extended to measure the control a group of vertices has on information flows through shortest paths between pairs of vertices. Let C be a subset of V. The GBC of C is defined as

$$GB(C) = \sum_{s,t \in V | s \neq t} \frac{\sigma_{st}(C)}{\sigma_{st}}, \tag{3.7}$$

where $\sigma_{st}(C)$ is the number of shortest paths joining s to t that pass through at least one vertex in C. Defining

$$\delta_s(C) = \sum_{t \in V | t \neq s} \frac{\sigma_{st}(C)}{\sigma_{st}}, \tag{3.8}$$

$GB(C)$ can be calculated as

$$GB(C) = \sum_{s \in V} \delta_s(C). \tag{3.9}$$

In a similar manner, the normalized GBC of a group of vertices C, denoted by

$NGB(C)$, is obtained by dividing $GB(C)$ by 2 $\binom{|V|}{2}$ and is interpreted as the probability that the information sent from a vertex s chosen uniformly at random from V to a different vertex t chosen uniformly at random from $V \setminus \{s\}$ passes through at least one vertex in C assuming that the information flows through one of the shortest paths joining s to t selected uniformly at random

In this thesis, we introduce a new BC measure called the k-step BC. BC measures how much control a vertex has on communications over a network. In several applications, it is not only important to get the information communicated but also the time when it is obtained. The k-step BC measures how likely it is for a vertex to get the information within the first k steps of the start of the communication. Similarly, the k-step GBC of a group of vertices measures the probability (after normalization) that the information is obtained by at least one group member within the first k steps of the start of the communication. From now on, we focus on the k-step GBC of groups of vertices. When the group size is one, this is translated into the k-step BC of a single vertex.

We denote by $\sigma_{st}^k(C)$ the total number of shortest paths joining s to t that pass through at least one vertex in C within the first k steps of the path. The k-step dependency of vertex s on group C, denoted by $\delta_s^k(C)$, is defined as

$$\delta_s^k(C) = \sum_{t \in V | t \neq s} \frac{\sigma_{st}^k(C)}{\sigma_{st}}. \tag{3.10}$$

After the relevant notation is introduced, we define $GB^k(C)$ mathematically as

$$GB^k(C) = \sum_{s,t \in V | s \neq t} \frac{\sigma_{st}^k(C)}{\sigma_{st}} \tag{3.11}$$

which can also be computed using k-step dependencies as

$$GB^k(C) = \sum_{s \in V} \delta_s^k(C). \tag{3.12}$$

For any positive integer k, we have that

$$GB^k(C) \leq GB^{k+1}(C) \leq GB^{md(G)}(C) = GB(C). \tag{3.13}$$

For any group C, the smallest positive integer k satisfying $GB^k(C) = GB(C)$ is called as the saturation number of C and is denoted by $s(C)$. Dividing $GB^k(C)$ by $2\binom{|V|}{2}$, we obtain the normalized k-step GBC of C denoted by $NGB^k(C)$. $NGB^k(C)$ represents the probability that at least one element of C gets the information, which is sent through a randomly chosen shortest path between randomly chosen pair of distinct vertices, within the first k steps of the start of the communication.

For a given graph $G = (V, E)$, group size g, and positive integer k, we denote by $NGB_g^{k*}(G)$ the maximum possible normalized k-step GBC value of a subset of V of size g. Mathematically, we have that

$$NGB_g^{k*}(G) = \max_{C \subseteq V : |C| = g} NGB^k(C). \tag{3.14}$$

For a given k and g, the group C of size g such that $NGB^k(C) = NGB_g^{k*}(G)$ is denoted by C_g^{k*}.

Similarly, we represent the maximum possible normalized GBC value of a subset of V of size g as $NGB_g^*(G)$ which can be calculated as

$$NGB_g^*(G) = \max_{C \subseteq V : |C| = g} NGB(C). \tag{3.15}$$

For a given g, the group C of size g such that $NGB(C) = NGB_g^*(G)$ is denoted by C_g^*.

CHAPTER 4

SOLUTION METHOD

In this chapter, we aim to develop an algorithm to compute the k-step GBC value for a given subset C of size g. Our aim here is to be able to compute the k-step GBC value of a large number of subsets of vertices. In real life social networks, there are tens of thousands of vertices and exact solution methods to find the group with the highest k-step GBC (or the classical GBC) value are not applicable, see e.g., the limitations of the integer programming formulation proposed in Chapter 7. The algorithm developed here can be used together with a sampling algorithm (e.g., [60], [54], [62]) that samples potentially good groups of vertices (in terms of the k-step GBC value). As the distribution of the (k-step) GBC values is right-skewed for several classes of graphs, one may need to sample several groups of vertices in order to find a group with a high (k-step) GBC value.

4.1 An Algorithm to Compute the k-step Group Betweenness Centrality

Brandes [4] proposes a method that computes the GBC of a single group in $O(nm)$ time. In cases where one wants to compute the GBC of a large number of groups, this method may not be very practical. Puzis et al. [8] propose another algorithm for the computation of the GBC. Assuming that the group sizes are equal to g, the algorithm has a preprocessing step that takes $O(n^3)$ time followed by GBC computations taking $O(g^3)$ time for each group. The time complexities of the algorithms of Brandes and Puzis et al. to compute the GBC of ℓ many groups each of size g is $O(\ell nm)$ and $O(n^3 + \ell g^3)$, respectively. When there are a large number of GBC computations and when $g \ll n$, the algorithm proposed by Puzis et al. performs better than that

proposed by Brandes. Moreover when $m = O(n^2)$, i.e., when the graph is dense, the time complexity of the algorithm of Brandes cannot be better than that of Puzis et al. for any ℓ and g values.

We compare the algorithms proposed by Brandes and Puzis et al. on a hi-tech computer firm network $G = (V, E)$ named as Hi-tech Network [71] which has 36 vertices and 91 edges. The network contains the friendship ties among the employees and it's giant connected part has 33 vertices and 91 edges. For different group sizes g, we enumerate all possible groups of size g and compute the GBC for each of them using both methods to find the group with the highest GBC value. For each group size g, the total computation time to compute the GBC of all groups of size g are given in Table 4.1 for both algorithms. For the algorithm proposed by Puzis et al., the preprocessing step takes less than 1 second and the group of size 6 with the highest GBC is computed within 47.63 seconds. On the other hand, it takes about 26 hours for the algorithm proposed by Brandes to evaluate the GBC of all groups of size 6. As the algorithm proposed by Puzis et al. is much faster, we utilize it as a backbone to develop an algorithm for the successive k-step GBC computations.

Table 4.1: The preprocessing and the total computational time (in seconds) to find a group of size g with the highest GBC using the algorithms proposed by Brandes and Puzis et al. on the Hi-tech network

Alg. \ **g**	**1**	**2**	**3**	**4**	**5**	**6**	**Prep.**	**Total**
Brandes	3.98	64.18	614.08	4477.44	30967.21	93564.43	0.00	129691.33
Puzis	0.00	0.01	0.07	0.74	6.20	47.63	0.14	54.79

* Prep.: Preprocessing time, Total: total computational time, Alg.: Algorithm

The algorithm proposed by Puzis et al. uses the notion of path betweenness centrality. For an ordered list of vertices $L = (x_1, x_2, \ldots, x_\ell)$, the path betweenness centrality of L, denoted by $PB(L)$, represents the total fraction of the shortest paths between pairs of vertices that pass through all of $x_1, x_2, \ldots, x_\ell$ in this order (some other vertices may be visited in between these vertices). Defining $\tilde{\sigma}_{st}(L)$ as the number of shortest paths

joining s to t that pass through all of $x_1, x_2, \ldots, x_\ell$ in this order, we have that

$$PB(L) = \sum_{s,t \in V | s \neq t} \frac{\tilde{\sigma}_{st}(L)}{\sigma_{st}}. \tag{4.1}$$

When L is a list of two vertices, say (x, y), we use the notation $PB(x, y)$ for $PB(L)$. The algorithm proposed by Puzis et al. uses path betweenness centrality of pairs of vertices only. For an ordered list of two vertices (x, y), we introduce k-step path betweenness centrality of (x, y), denoted by $PB^k(x, y)$. $PB^k(x, y)$ represents the total fraction of shortest paths that pass through x and y in this order both within k steps and is mathematically defined in Equation 4.2.

$$PB^k(x, y) = \sum_{s,t \in V | s \neq t} \frac{\tilde{\sigma}^k_{st}(x, y)}{\sigma_{st}}. \tag{4.2}$$

Here $\tilde{\sigma}^k_{st}(x, y)$ is the total number of shortest paths joining s to t that pass through first vertex x and then y both within k steps. We can rewrite $\tilde{\sigma}^k_{st}(x, y)$ as

$$\tilde{\sigma}^k_{st}(x, y) = \sigma^k_{st}(y) \frac{\sigma_{sy}(x)}{\sigma_{sy}}. \tag{4.3}$$

Here $\sigma^k_{st}(y)$ is the number of shortest paths joining s to t that pass through y within the first k steps of the path. We multiply $\sigma^k_{st}(y)$ by the fraction of the shortest paths joining s to y that pass through x to obtain $\tilde{\sigma}^k_{st}(x, y)$. When we substitute $\tilde{\sigma}^k_{st}(x, y)$ in Equation 4.2, we get

$$PB^k(x, y) = \sum_{s \in V} \sum_{t \in V | t \neq s} \frac{\sigma^k_{st}(y)}{\sigma_{st}} \frac{\sigma_{sy}(x)}{\sigma_{sy}}, \tag{4.4}$$

which is simplified to

$$PB^k(x, y) = \sum_{s \in V} \delta^k_s(y) \frac{\sigma_{sy}(x)}{\sigma_{sy}}. \tag{4.5}$$

We define the k-step path betweenness centrality matrix PB^k as $[PB^k]_{x,y \in V}$. Using Equation 4.5, this matrix can be computed in $O(n^3)$ time. Note that $PB^k(x, x) =$

$GB^k(x)$ for any $x \in V$. For any $M \subseteq V$, $PB^k_M(x,y)$ represents the k-step path betweenness centrality of (x,y) disregarding the shortest paths that pass through any vertex in M within the first k steps of the path. Mathematically we have

$$PB^k_M(x,y) = \sum_{s,t \in V | s \neq t} \frac{\tilde{\sigma}^{k,M}_{st}(x,y)}{\sigma_{st}}, \tag{4.6}$$

where $\tilde{\sigma}^{k,M}_{st}(x,y)$ is the number of shortest paths joining s to t that pass through first vertex x and then y both within k steps and that do not pass through any vertex in M within the first k steps. Moreover, we define σ^M_{st} as the number of shortest paths joining s to t that do not pass through any vertex in M.

Now, we describe an algorithm that computes the k-step group betweenness centrality of a group of vertices using path betweenness which is a modification of the algorithm proposed by Puzis et al. [8] for the computation of the GBC of a group of vertices. The steps of the algorithm are given in Algorithm 1. The inputs of the algorithm are a graph $G = (V, E)$, a group of vertices $C = \{c_1, c_2, \ldots, c_g\}$, and a positive integer k and the output is $GB^k(C)$. Algorithm 1 has a preprocessing step where three matrices σ, d, and PB^k are computed. This step takes $O(n^3)$ time and does not depend on the group size g. During the algorithm, the contributions of elements of C on $GB^k(C)$ are considered one by one between lines 8 and 27. A set M is defined in line 4 which is initially an empty set. Once their contributions to $GB^k(C)$ are considered, the elements of C are added one by one to M during the algorithm.

We define two $g \times g$ matrices $\sigma^M = [\sigma^M_{xy}]_{x,y \in C}$ and $PB^k_M = [PB^k_M(x,y)]_{x,y \in C}$. Initially in lines 5 and 6, we have that $\sigma^M_{xy} = \sigma_{xy}$ and $PB^k_M(x,y) = PB^k(x,y)$, $\forall x, y \in C$ as M is the empty set for now. Moreover $GB^k(C)$ is initialized to zero in line 7 as the contributions of the elements of C are not yet considered. In each iteration within the for loop between lines 8 and 27, an element of $C \setminus M$, say c_i, is taken and its contribution to $GB^k(C)$ is considered. This is done in line 9 by the equation $GB^k(M \cup \{c_i\}) = GB^k(M) + PB^k_M(c_i, c_i)$. As M will be updated to $M \cup \{c_i\}$, to be able to determine the contribution of the following elements of C to $GB^k(C)$, the matrix PB^k_M has to be updated. For this purpose, we first update the matrix σ^M between lines 11 and 13. For any $x, y \in C$, σ^M_{xy} is subject to change when c_i lies on at least one shortest path joining x to y. In this case, $\sigma^{M \cup \{c_i\}}_{xy}$ becomes

$\sigma^M_{xy} - \sigma^M_{xc_i}\sigma^M_{c_iy}$, where $\sigma^M_{xc_i}\sigma^M_{c_iy}$ is the number of shortest paths joining x to y that do not pass through any element of M, but that pass through c_i. Between lines 14 and 24, the matrix PB^k_M is updated. In lines 14 and 15, $PB^k_{M\cup\{c_i\}}(x,x)$ is computed for any $x \neq c_i$ by the equation $PB^k_{M\cup\{c_i\}}(x,x) = PB^k_M(x,x) - PB^k_M(c_i,x) - PB^k_M(x,c_i)$. The case where $x = y \neq c_i$ is called as case 1. If we are not in case 1 and if x, y, and c_i all lie on at least one shortest path in the graph, then $PB^k_{M\cup\{c_i\}}(x,y)$ is computed between lines 16 and 21. In case 2, we are not in case 1 and x lies on a shortest path joining c_i to y. In this case, $PB^k_{M\cup\{c_i\}}(x,y)$ is computed by subtracting from $PB^k_M(x,y)$ the k-step path betweenness centrality of (c_i, y) disregarding the shortest paths that pass through any vertex in M within the first k steps of the path multiplied with $\sigma^M_{c_ix}\sigma^M_{xy}/\sigma^M_{c_iy}$ which is the fraction of the shortest paths (not going through any vertex of M) joining c_i to y that pass through x. If c_i lies on a shortest path joining x to y (case 3), $PB^k_{M\cup\{c_i\}}(x,y)$ is computed in line 19. And if y lies on a shortest path joining x to c_i (case 4), $PB^k_{M\cup\{c_i\}}(x,y)$ is computed in line 21. If x, y, and c_i do not all lie on any shortest path in the graph, then $PB^k_{M\cup\{c_i\}}(x,y)$ will be the same as $PB^k_M(x,y)$ as in line 23. Finally, in line 26, M is updated as $M \cup \{c_i\}$.

The preprocessing step of Algorithm 1 takes $O(n^3)$ time. For each $i \in \{1, 2, \ldots g\}$, the algorithm spends $O(g^2)$ time within the for loop between lines 10 and 25. Therefore $O(g^3)$ time is spent after the preprocessing step. Note that the preprocessing step is to be performed only once when one wants to evaluate the k-step group betweenness centrality for several subsets of vertices of a graph. Therefore the k-step GBC for ℓ many groups each of size g can be evaluated in $O(n^3 + \ell g^3)$ time. Note also that the preprocessing step does not depend on the group size g. After the preprocessing step is done, one can evaluate the k-step GBC for groups having different sizes as well.

Algorithm 1 k-step GBC

Inputs: $G = (V, E), C = \{c_1, c_2, \ldots, c_g\} \subseteq V$, and a positive integer k

Output: $GB^k(C)$

Preprocessing: Compute σ, d, and PB^k.

$M \leftarrow \{\}$

$\sigma^M_{xy} \leftarrow \sigma_{xy}, \forall x, y \in C$

$PB^k_M(x, y) \leftarrow PB^k(x, y), \forall x, y \in C$

$GB^k(C) \leftarrow 0$

for $i = 1$ to g **do**

  $GB^k(C) \leftarrow GB^k(C) + PB^k_M(c_i, c_i)$

  for $\forall x, y \in C$ **do**

    if $d(x, y) = d(x, c_i) + d(c_i, y)$ **then**

      $\sigma^{M\cup\{c_i\}}_{xy} \leftarrow \sigma^M_{xy} - \sigma^M_{xc_i}\sigma^M_{c_iy}$

    end if

    if $x = y \neq c_i$ **then**

      $PB^k_{M\cup\{c_i\}}(x, x) \leftarrow PB^k_M(x, x) - PB^k_M(c_i, x) - PB^k_M(x, c_i)$

    else if $d(c_i, y) = d(c_i, x) + d(x, y)$ **then**

      $PB^k_{M\cup\{c_i\}}(x, y) \leftarrow PB^k_M(x, y) - \frac{\sigma^M_{c_ix}\sigma^M_{xy}PB^k_M(c_i,y)}{\sigma^M_{c_iy}}$

    else if $d(x, y) = d(x, c_i) + d(c_i, y)$ **then**

      $PB^k_{M\cup\{c_i\}}(x, y) \leftarrow PB^k_M(x, y) - \frac{\sigma^M_{xc_i}\sigma^M_{c_iy}PB^k_M(x,y)}{\sigma^M_{xy}}$

    else if $d(x, c_i) = d(x, y) + d(y, c_i)$ **then**

      $PB^k_{M\cup\{c_i\}}(x, y) \leftarrow PB^k_M(x, y) - \frac{\sigma^M_{xy}\sigma^M_{yc_i}PB^k_M(x,c_i)}{\sigma^M_{xc_i}}$

    else

      $PB^k_{M\cup\{c_i\}}(x, y) \leftarrow PB^k_M(x, y)$

    end if

  end for

  $M \leftarrow M \cup \{c_i\}$

end for

CHAPTER 5

ILLUSTRATIVE EXAMPLE

In this chapter, we illustrate k-step path betweenness and k-step GBC calculations on a graph $G = (V, E)$ of order 6 given in Figure 5.1.

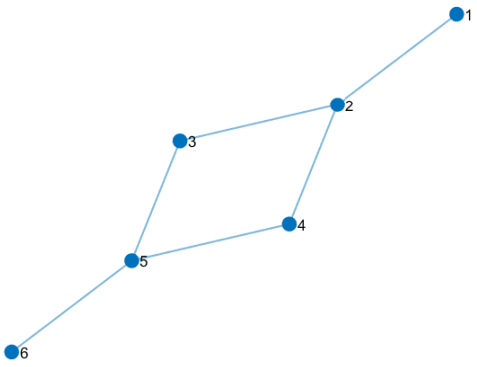


Figure 5.1: A graph of order $n = 6$

For this graph, distance (d), sigma (σ), and dependency (δ) matrices are calculated as

$$d = \begin{pmatrix} 0 & 1 & 2 & 2 & 3 & 4 \\ 1 & 0 & 1 & 1 & 2 & 3 \\ 2 & 1 & 0 & 2 & 1 & 2 \\ 2 & 1 & 2 & 0 & 1 & 2 \\ 3 & 2 & 1 & 1 & 0 & 1 \\ 4 & 3 & 2 & 2 & 1 & 0 \end{pmatrix}, \sigma = \begin{pmatrix} 1 & 1 & 1 & 1 & 2 & 2 \\ 1 & 1 & 1 & 1 & 2 & 2 \\ 1 & 1 & 1 & 2 & 1 & 1 \\ 1 & 1 & 2 & 1 & 1 & 1 \\ 2 & 2 & 1 & 1 & 1 & 1 \\ 2 & 2 & 1 & 1 & 1 & 1 \end{pmatrix}, \delta = \begin{pmatrix} 5 & 5 & 2 & 2 & 2 & 1 \\ 1 & 5 & 2 & 2 & 2 & 1 \\ 1 & 2.5 & 5 & 1 & 2.5 & 1 \\ 1 & 2.5 & 1 & 5 & 2.5 & 1 \\ 1 & 2 & 2 & 2 & 5 & 1 \\ 1 & 2 & 2 & 2 & 5 & 5 \end{pmatrix}.$$

Note that using the matrices δ and d, the matrix δ^k can be computed. For any two vertices $s, v \in V$ and any positive integer k, we have that $\delta_s^k(v) = \delta_s(v)$ if $d(s,v) \leq k$ and $\delta_s^k(v) = 0$ otherwise. Consider the subset $C = \{3, 6\}$ of V. Utilizing Equation 4.5, we have that

$$PB^1(3,6) = \sum_{s \in V} \delta_s^1(6) \frac{\sigma_{s6}(3)}{\sigma_{s6}} = 0,$$

$$PB^2(3,6) = \sum_{s \in V} \delta_s^2(6) \frac{\sigma_{s6}(3)}{\sigma_{s6}} = \delta_3^2(6) \frac{\sigma_{36}(3)}{\sigma_{36}} = 1,$$

$$PB^3(3,6) = \sum_{s \in V} \delta_s^3(6) \frac{\sigma_{s6}(3)}{\sigma_{s6}} = \delta_3^3(6) \frac{\sigma_{36}(3)}{\sigma_{36}} + \delta_2^3(6) \frac{\sigma_{26}(3)}{\sigma_{26}} = 1.5,$$

$$PB^4(3,6) = \sum_{s \in V} \delta_s^4(6) \frac{\sigma_{s6}(3)}{\sigma_{s6}} = \delta_3^4(6) \frac{\sigma_{36}(3)}{\sigma_{36}} + \delta_2^4(6) \frac{\sigma_{26}(3)}{\sigma_{26}} + \delta_1^4(6) \frac{\sigma_{16}(3)}{\sigma_{16}} = 2.$$

$PB^1(3,6) = 0$ means that there is no shortest path joining two distinct vertices in G that passes through first vertex 3 and then vertex 6 both within 1 step. Computing the k-step path betweenness centrality for every ordered pair of vertices (x, y), we obtain the following k-step path betweenness centrality matrices $PB^k, k = 1, 2, 3, 4$.

$$PB^1 = \begin{pmatrix} 6 & 5 & 0 & 0 & 0 & 0 \\ 1 & 15 & 2 & 2 & 0 & 0 \\ 0 & 2.5 & 9 & 0 & 2.5 & 0 \\ 0 & 2.5 & 0 & 9 & 2.5 & 0 \\ 0 & 0 & 2 & 2 & 15 & 1 \\ 0 & 0 & 0 & 0 & 5 & 6 \end{pmatrix} \qquad PB^2 = \begin{pmatrix} 8 & 5 & 2 & 2 & 0 & 0 \\ 3 & 17 & 4.5 & 4.5 & 2 & 0 \\ 1 & 3.5 & 14 & 1 & 3.5 & 1 \\ 1 & 3.5 & 1 & 14 & 3.5 & 1 \\ 0 & 2 & 4.5 & 4.5 & 17 & 3 \\ 0 & 0 & 2 & 2 & 5 & 8 \end{pmatrix}$$

$$PB^3 = \begin{pmatrix} 9 & 5 & 2 & 2 & 2 & 0 \\ 4 & 19 & 4.5 & 4.5 & 4 & 1 \\ 1.5 & 4.5 & 14 & 1 & 4.5 & 1.5 \\ 1.5 & 4.5 & 1 & 14 & 4.5 & 1.5 \\ 1 & 4 & 4.5 & 4.5 & 19 & 4 \\ 0 & 2 & 2 & 2 & 5 & 9 \end{pmatrix} \qquad PB^4 = \begin{pmatrix} 10 & 5 & 2 & 2 & 2 & 1 \\ 5 & 19 & 4.5 & 4.5 & 4 & 2 \\ 2 & 4.5 & 14 & 1 & 4.5 & 2 \\ 2 & 4.5 & 1 & 14 & 4.5 & 2 \\ 2 & 4 & 4.5 & 4.5 & 19 & 5 \\ 1 & 2 & 2 & 2 & 5 & 10 \end{pmatrix}$$

Note that $PB^k = PB^4$ for every positive integer $k \geq 4$ as $md(G) = 4$ and $PB^{md(G)}$ is a symmetric matrix. For a vertex $x \in V$, the diagonal element $P^k(x, x)$ of the k-step path betweenness centrality matrix P^k is equal to $GB^k(\{x\})$ which is the k-step betweenness centrality of x. When we investigate the saturation numbers of groups consisting of a single vertex, we obtain that $s(\{1\}) = s(\{6\}) = 4$, $s(\{3\}) = s(\{4\}) = 2$, and $s(\{2\}) = s(\{5\}) = 3$.

We now illustrate the calculation of the 2-step GBC of a subset $C = \{c_1, c_2, c_3\}$ of vertices, where $c_1 = 1, c_2 = 6$, and $c_3 = 5$, using Algorithm 1. The matrices d, σ, and PB^2 are already computed. The set M is initialized to $\emptyset$. σ^M_{xy} and $PB^k_M(x, y)$ are obtained using the equations in lines 5 and 6 of Algorithm 1 for every $x, y \in C$ as

$$\sigma^{\{\}} = \begin{pmatrix} 1 & 2 & 2 \\ 2 & 1 & 1 \\ 2 & 1 & 1 \end{pmatrix}, \qquad PB^2_{\{\}} = \begin{pmatrix} 8 & 0 & 0 \\ 0 & 8 & 5 \\ 0 & 3 & 17 \end{pmatrix}.$$

We next initialize $GB^2(C)$ to zero. $PB^2_M(c_i, c_i)$ represents the contribution of vertex c_i to $GB^2(C)$ for any $c_i \notin M$. We first consider adding vertex 1 to M and obtain that $GB^2(C) = PB^2_M(1, 1) = 8$. We next update the matrices σ^M and PB^2_M following the computations given between lines 11 and 13 and between lines 14 and 24 of Algorithm 1, respectively. The updated matrices are given below.

$$\sigma^{\{1\}} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} \qquad PB^2_{\{1\}} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 8 & 5 \\ 0 & 3 & 17 \end{pmatrix}$$

Secondly, vertex 6 is added to M. Therefore we get that $GB^2(C) = 8 + PB^2_{\{1\}}(6, 6) = 16$ and we update M to $\{1, 6\}$. Moreover the matrices σ^M and PB^2_M are updated as

$$\sigma^{\{1,6\}} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad PB^2_{\{1,6\}} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 9 \end{pmatrix}.$$

Finally, vertex 5 is added to M and we obtain that $GB^2(C) = 16 + PB^2_{\{1,6\}}(5, 5) =$

25. Now, the contributions of all elements of C to $GB^2(C)$ have been accumulated. When we finally update the matrices σ^M and PB^2_M, both become the matrix of all zeroes. When we compute the normalized 2-step GBC of C, we obtain that $NGB^2(C) = GB^2(C)/(2\binom{|V|}{2}) = 25/30 = 0.833$. This value represents the probability that at least one element of C gets the information, which is sent through a randomly chosen shortest path between randomly chosen pair of distinct vertices, within the first 2 steps of the start of the communication.

CHAPTER 6

COMPUTATIONAL EXPERIMENTS

In this section, we perform several computational experiments on real and randomly generated networks to

- understand how $NGB_g^{k*}(G)$ change as k and g change,
- understand how $NGB_g^{k*}(G)$ change as the density of the graphs change (in random networks),
- understand the potential benefits of using k-step GBC instead of GBC as a centrality measure to access more of the information communicated within k steps,
- understand the limitations of Algorithm 1 in finding the group of size g with the highest k-step GBC value,
- understand the change in running time of Algorithm 1 as k and g change.

All computational experiments are implemented in MATLAB and executed on a computer with Intel(R) Core(TM) i7-6500U CPU with 2.5GHz speed and 8 GB RAM.

6.1 The Will57 Network

The first network we consider is a semiconductor device problem network called the Will57 network [72] that has 57 nodes and 127 edges. Figure 6.1 graphically displays this network.

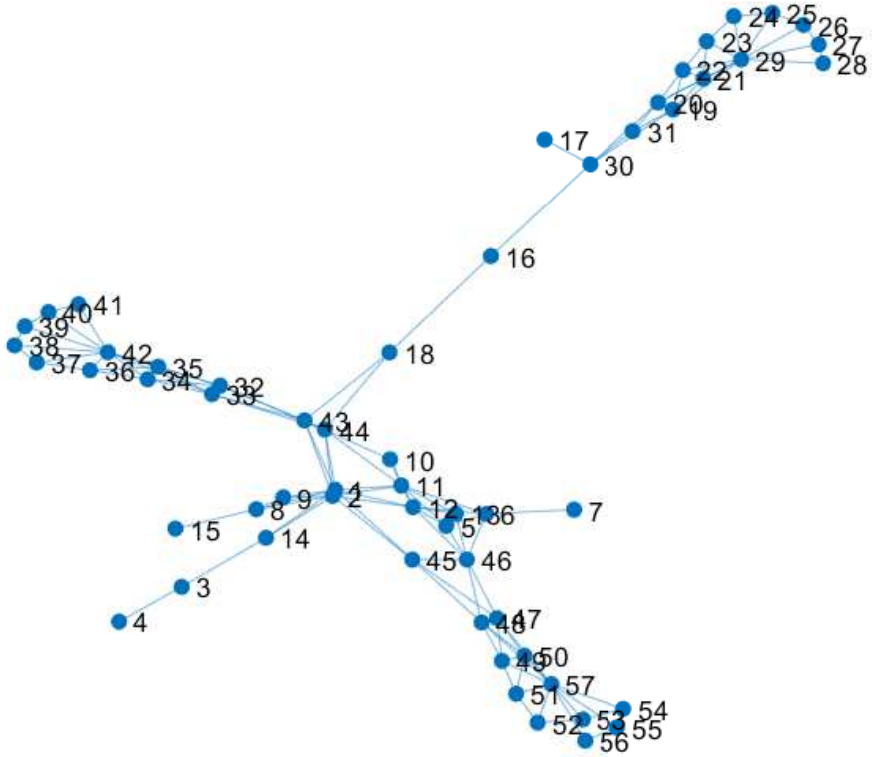


Figure 6.1: The Will57 network

First, denoting the Will57 network by G, we find $NGB_g^{k*}(G)$ values for different k and g values. These values are shown in Table 6.1. For example, when $k = 1$ and $g = 1$, there exists a vertex (vertex 57) in G that gets the information communicated (between a random pair of distinct vertices through a random shortest path) within 1 step with probability 0.11. On the other hand, when $k = 3$ and $g = 4$, there exists a group of 4 vertices (the vertices $\{30, 43, 44, 45\}$) in G that gets the information communicated within 3 steps with probability 0.75. When we examine Table 6.1 row by row, we see that for a fixed g, $NGB_g^{k*}(G)$ values increase as k increases until some k value and remain constant thereafter. This k value is called as the saturation number of the graph G with respect to group size g and is denoted by $s_g(G)$. Mathematically, we define $s_g(G)$ as the minimum k value such that $NGB_g^{k*}(G) = NGB_g^*(G)$. For the Will57 network, we have that $s_g(G) = 6$ for every $g \in \{1, 2, 3, 4, 5\}$. When we examine Table 6.1 column by column, we see that for a fixed k, $NGB_g^{k*}(G)$ values increase as g increases as expected.

Table 6.1: $NGB_g^{k*}(G)$ for different k and g values

g \ k	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
1	0.11	0.15	0.24	0.30	0.39	0.42	0.42	0.42	0.42	0.42	0.42
2	0.23	0.28	0.45	0.52	0.60	0.66	0.66	0.66	0.66	0.66	0.66
3	0.34	0.40	0.63	0.68	0.71	0.75	0.75	0.75	0.75	0.75	0.75
4	0.43	0.51	0.75	0.77	0.78	0.79	0.79	0.79	0.79	0.79	0.79
5	0.50	0.61	0.81	0.83	0.84	0.85	0.85	0.85	0.85	0.85	0.85

Table 6.2: $NGB^k(C_g^*)$ for different k and g values

g \ k	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
1	0.04	0.08	0.19	0.29	0.39	0.42	0.42	0.42	0.42	0.42	0.42
2	0.11	0.24	0.40	0.49	0.58	0.66	0.66	0.66	0.66	0.66	0.66
3	0.23	0.36	0.53	0.62	0.67	0.75	0.75	0.75	0.75	0.75	0.75
4	0.20	0.39	0.61	0.68	0.72	0.79	0.79	0.79	0.79	0.79	0.79
5	0.30	0.51	0.74	0.83	0.84	0.85	0.85	0.85	0.85	0.85	0.85

Second, for each group size g, we find the group C_g^* that has the highest normalized GBC value, i.e., the group C_g^* satisfying $NGB(C_g^*) = NGB_g^*(G)$. For the group C_g^*, we compute $NGB^k(C_g^*)$ for different k values and report them in Table 6.2. For example, $NGB^3(C_4^*) = 0.61$ which means that the group C_4^* gets the information communicated within 3 steps with probability 0.61. Assume that it is critical for a user to obtain the information communicated within 3 steps. If s/he chooses the group (of size 4) according to the maximum GBC value, s/he would choose C_4^* which is $\{43, 44, 45, 46\}$ and can get the information within 3 steps with probability 0.61. On the other hand, if the user chooses the group according to the maximum 3-step GBC value, s/he would choose C_4^{3*} which is $\{30, 43, 44, 45\}$ and can get the information within 3 steps with probability 0.75. In other words, replacing the vertex 46 with 30, the user gets a chance to obtain the information faster with a larger probability.

Therefore, if the user aims to obtain the information within 3 steps with a higher probability, it is better to use the 3-step GBC measure instead of the classical GBC measure.

Third, we fix the group size g to 2. In this case, we have that $C_2^* = \{43, 44\}$ and $NGB(C_2^*) = 0.66$. We find C_2^{k*} for different k values and show them in Figure 6.2. Members of different optimum groups for different k values are remarked with different symbols in Figure 6.2. We have that $C_2^{1*} = \{29, 42\}$ and elements of this group are remarked with triangles. Moreover $NGB_2^{1*} = 0.23$ which means that within 1 step the information can be obtained with a maximum probability of 0.23 by the elements of the group C_2^{1*}. When k becomes 2 or 3, the optimum groups C_2^{2*} and C_2^{3*} are the same and are equal to $\{30, 44\}$. Furthermore, we have that $NGB_2^{2*} = 0.28$ and $NGB_2^{3*} = 0.45$. In other words, the optimum group for $k = 2$ or 3 gets the information communicated within 2 steps with probability 0.28 and within 3 steps with probability 0.45. When k becomes 4, the optimum group C_2^{4*} is $\{16, 45\}$ and NGB_2^{4*} is equal to 0.52. When k becomes 5, the optimum group C_2^{5*} is $\{30, 45\}$ which has a 5-step normalized GBC of 0.60 and finally when k becomes 6, the optimum group becomes the same as C_2^* which is $\{43, 44\}$. This example shows that the optimum group maximizing the k-step GBC value may change from k to k. For this example, the optimum group moves toward the middle of the graph starting from outer vertices as k increases.

Forth, for the group size $g = 3$, we examine the distribution of the k-step normalized GBC values of all subsets of V of size 3. For each k value, we compute $NGB^k(C)$ for every subset C of V of size 3 and make a histogram of the $NGB^k(C)$ values (see Figure 6.3). Moreover, for each k value, Table 6.3 reports the average, median, and maximum $NGB^k(C)$ values on the Will57 network. It can be seen from Figure 6.3 and Table 6.3 that the distribution of the $NGB^k(C)$ values is right-skewed. Therefore, one may need to sample several groups to obtain a group that has a normalized k-step GBC value that is close to the optimum. The algorithm proposed by Puzis et al. and Algorithm 1 serve this purpose as once the preprocessing step is performed, these algorithms can evaluate the GBC and k-step GBC, respectively, of several groups successively very fast.

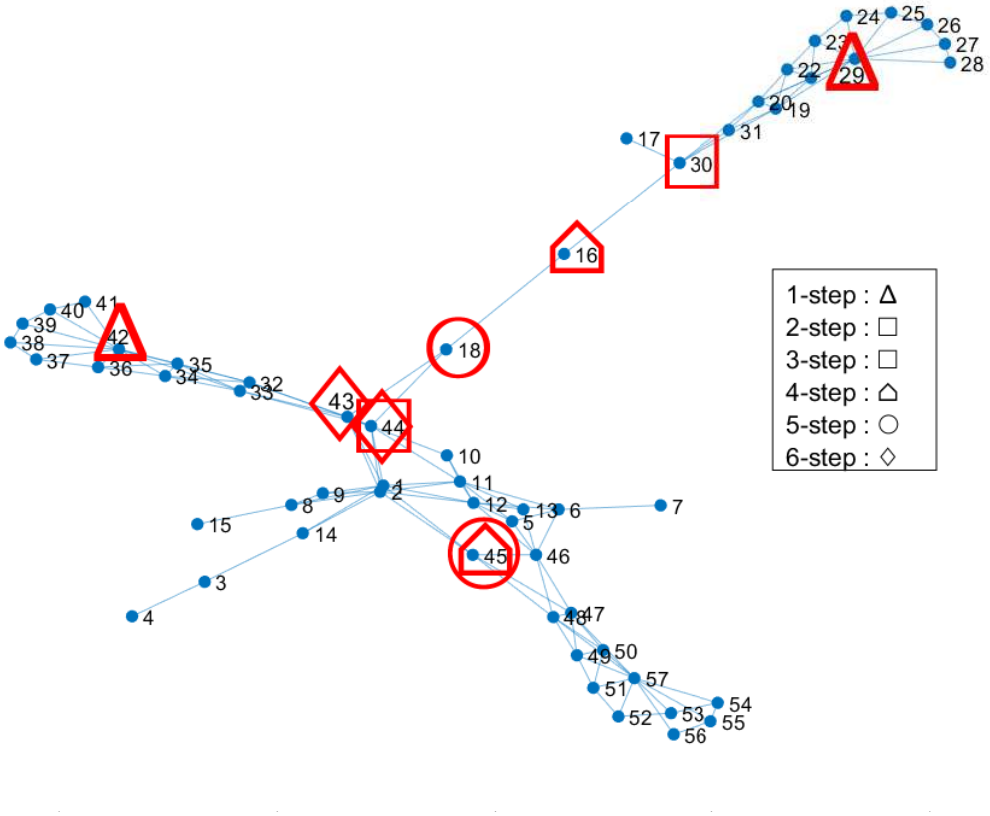


Figure 6.2: Groups of size 2 maximizing the k-step normalized GBC for different k values on the Will57 network

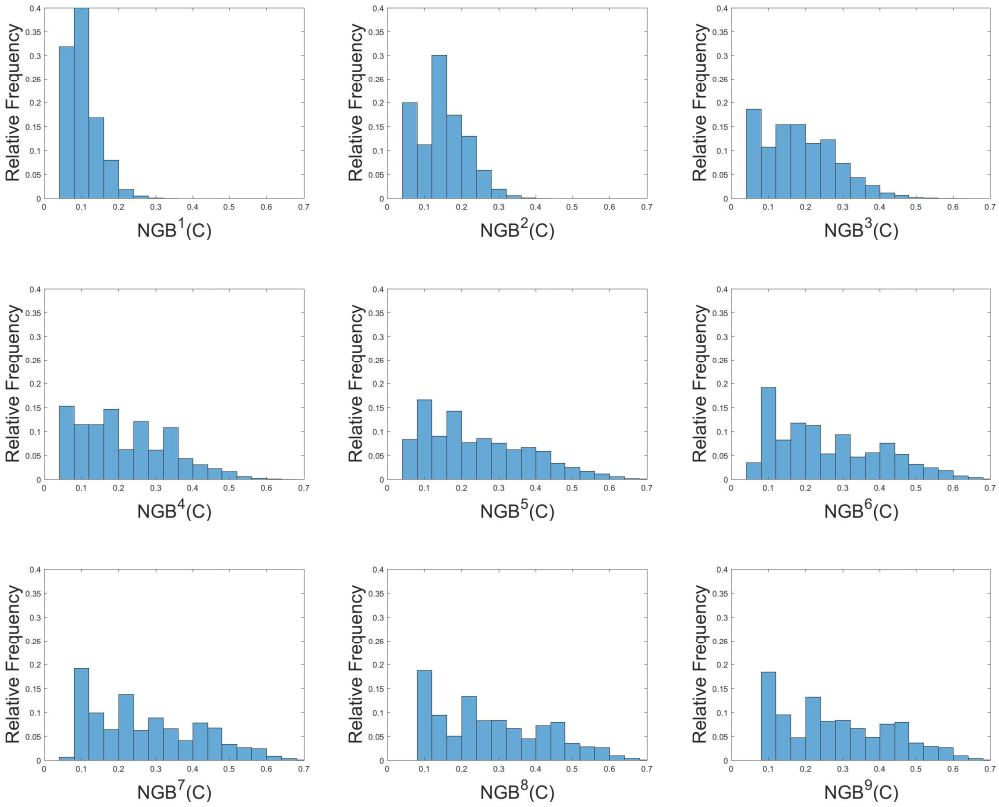


Figure 6.3: Distribution of the $NGB^k(C)$ values of all subsets C of V of size 3 on the Will57 network

Table 6.3: Average, median, and maximum $NGB^k(C)$ of all subsets C of V of size 3 on the Will57 network

k	Median $NGB^k(C)$	Average $NGB^k(C)$	Maximum $NGB^k(C)$
1	0.09	0.10	0.34
2	0.14	0.15	0.40
3	0.18	0.19	0.63
4	0.19	0.22	0.68
5	0.21	0.24	0.71
6	0.22	0.26	0.75
7	0.24	0.27	0.75
8	0.25	0.28	0.75
9	0.26	0.28	0.75

Now, we report the computation time required for Algorithm 1 to compute $NGB^k(C)$ values for all subsets C of V of size 1 to 5 in Table 6.4.

Table 6.4: The preprocessing and the total computational time (in s) to compute $NGB^k(C)$ values for all possible groups C of size g on the Will57 network

	Preprocessing	**Time (in s) to Compute $NGB^k(C)$ Values for All Possible Groups C of Size g**				
	Time (in s)	**Group Size g**				
k	**to Compute PB^k**	**1**	**2**	**3**	**4**	**5**
1	0.009	0.00	0.02	0.36	6.62	127.99
2	0.009	0.00	0.02	0.49	9.75	140.31
3	0.009	0.00	0.03	0.53	9.74	149.65
4	0.009	0.00	0.04	0.50	9.71	156.45
5	0.010	0.00	0.04	0.69	10.98	141.07
6	0.010	0.00	0.02	0.45	10.28	151.10
7	0.011	0.00	0.05	0.52	10.20	159.39
8	0.012	0.00	0.04	0.56	11.10	170.97
9	0.012	0.00	0.04	0.62	11.29	170.49
10	0.012	0.00	0.04	0.53	11.54	172.64

In the preprocessing step of Algorithm 1, three matrices d, σ, and PB^k are computed. The first two of these matrices are independent of k and the group size g. In our implementation, Algorithm 1 computes the matrices d and σ in 0.18 seconds. The matrix PB^k is computed once for each k value. The time to compute PB^k for each k value is given in Table 6.4 in the column titled as "Preprocessing Time (in s) to compute PB^k". Note that the actual total preprocessing time to compute all the values in the table is the sum of the values in the column titled "Preprocessing Time (in s) to compute PB^k" plus 0.18 seconds. Note that as k increases, the preprocessing time to compute PB^k slightly increases. The computation time (in s) of $NGB^k(C)$ values for all possible groups C of size g is given in the following columns. For example, when $k = 7$ and $g = 5$, the time to compute the $NGB^k(C)$ values for all $\binom{|V|}{5}$ many possible groups C of size 5 is equal to 141.07 seconds. Observe that for a fixed g, as

k increases, the time to compute all $NGB^k(C)$ values increases as well. Moreover, for a fixed k, as g increases, the time to compute all $NGB^k(C)$ values also increases. Note, however, that this increase is partially attributed to the increase in the number of possible groups. When we find the average time to compute one $NGB^k(C)$ value, this value turns out to be between 1.25×10^{-5} and 4.12×10^{-5} for every pair of k and g values for the Will57 network.

6.2 Cheminformatics Networks

We secondly consider 10 real-life cheminformatics (enzymes) networks [71]. These networks are chosen starting from the top of a list of networks provided in `https://www.networkrepository.com/chem` that have between 30 and 70 vertices. Some properties of the chosen 10 networks are given in Table 6.5.

Table 6.5: Some properties of the considered 10 cheminformatics networks.

Network	$\|V\|$	$\|E\|$	**md**	**ad**	d_{max}	d_{avg}	**Density**
g10	32	53	19	7.26	5	3.31	0.11
g104	32	64	11	4.64	5	4.00	0.13
g105	33	69	12	4.67	7	4.18	0.13
g1	37	84	12	4.98	7	4.54	0.13
g116	42	74	14	5.57	5	3.52	0.09
g13	42	75	23	8.56	5	3.57	0.09
g102	42	82	24	8.71	6	3.91	0.10
g101	45	88	15	5.84	7	3.91	0.09
g112	51	95	15	6.19	5	3.73	0.07
g103	59	115	13	4.92	9	3.90	0.07

* md: maximum distance, ad: average distance, d_{max}: maximum degree, d_{avg}: average degree.

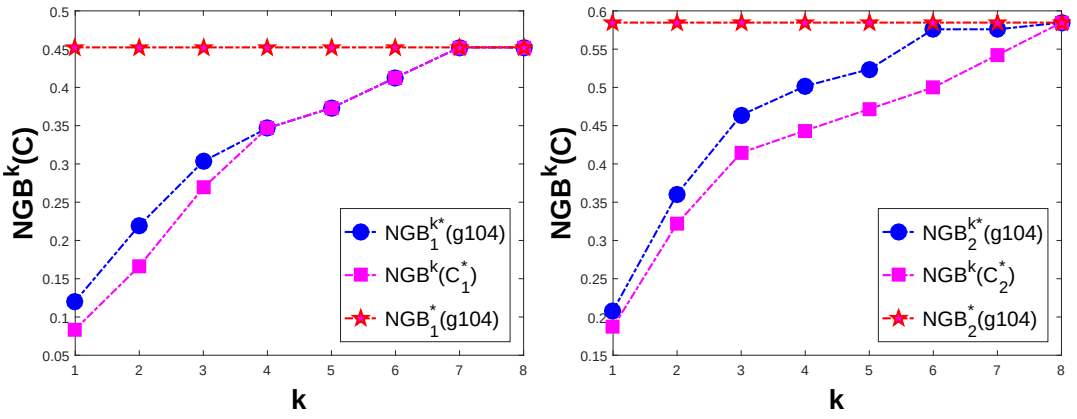
Table 6.6 reports $NGB_g^{k*}(G)$ and $NGB^k(C_g^*)$ for different combinations of k and g values for the considered cheminformatics networks. For any network G and group size g, $NGB_g^{k*}(G)$ and $NGB^k(C_g^*)$ are non-decreasing in k. From Table 6.6, we can see that $NGB_g^{k*}(G)$ and $NGB^k(C_g^*)$ values increase with decreasing marginal

changes until some k value and remain constant thereafter. Consider, for example, the network g13. When the group size is 4, there is a 0.38 increase in the optimum k-step GBC value as k increases from 1 to 3, i.e., $NGB_4^{3*}(g13) - NGB_4^{1*}(g13) = 0.38$. As k increases from 3 to 5, the increase in the optimum k-step GBC value drops to 0.17. Furthermore, we have that the saturation number of $g13$ with respect to group size 4 is equal to 5, i.e., $s_4(g13) = 5$, implying that $NGB_4^{k*}(g13) = 0.85$ for all $k \geq 5$ values.

For the network $g104$, Figure 6.4 shows how $NGB_g^{k*}(g104)$ (line with circle symbol) and $NGB^k(C_g^*)$ (line with square symbol) values change as k increases for different group sizes. Consider the group size $g = 2$. When $k = 1$, $NGB_2^{1*}(g104)$ is 0.021 bigger than $NGB^1(C_2^*)$, i.e., the group C_2^{1*} has a 0.021 larger probability of getting the information within 1 step than the group C_2^*. The difference between $NGB_2^{k*}(g104)$ and $NGB^k(C_2^*)$ increases to 0.076 when k becomes 6. This example shows that, even though $NGB_g^{k*}(G) \geq NGB^k(C_g^*)$ for every k and $NGB_g^{k*}(G) = NGB^k(C_g^*)$ for large enough k values, the difference $NGB_g^{k*}(G) - NGB^k(C_g^*)$ is not necessarily decreasing in k.

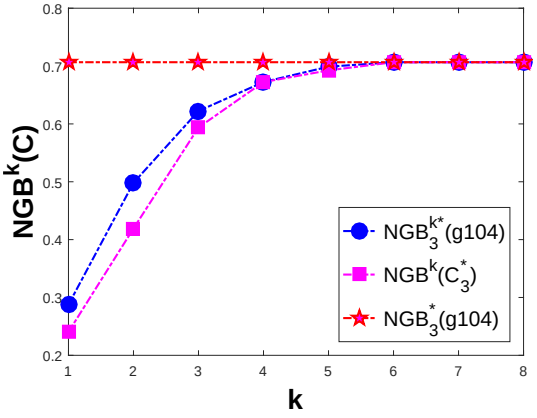
Table 6.6: $NGB_g^{k*}(G)$ and $NGB^k(C_g^*)$ for different k and g values on the considered 10 cheminformatics networks

		$\boldsymbol{NGB_g^{k*}(G)}$				$\boldsymbol{NGB^k(C_g^*)}$			
Network	**Group Size g**	**k=1**	**k=3**	**k=5**	**k=7**	**k=1**	**k=3**	**k=5**	**k=7**
g10	2	0.22	0.43	0.61	0.73	0.17	0.37	0.56	0.73
	4	0.42	0.78	0.86	0.86	0.35	0.76	0.86	0.86
g104	2	0.21	0.46	0.52	0.58	0.19	0.41	0.47	0.54
	4	0.36	0.73	0.78	0.78	0.32	0.73	0.78	0.78
g105	2	0.21	0.41	0.51	0.58	0.13	0.33	0.51	0.58
	4	0.37	0.68	0.76	0.76	0.34	0.65	0.75	0.76
g1	2	0.25	0.62	0.68	0.68	0.23	0.62	0.68	0.68
	4	0.41	0.75	0.80	0.80	0.34	0.75	0.80	0.80
g116	2	0.15	0.35	0.48	0.63	0.13	0.32	0.46	0.61
	4	0.29	0.67	0.85	0.85	0.26	0.60	0.85	0.85
g13	2	0.15	0.36	0.48	0.63	0.14	0.32	0.48	0.63
	4	0.30	0.68	0.85	0.85	0.28	0.68	0.85	0.85
g102	2	0.17	0.39	0.55	0.62	0.16	0.39	0.55	0.62
	4	0.32	0.63	0.75	0.75	0.27	0.57	0.74	0.75
g101	2	0.18	0.38	0.60	0.62	0.14	0.37	0.56	0.62
	4	0.32	0.61	0.76	0.77	0.27	0.59	0.72	0.77
g112	2	0.13	0.42	0.62	0.64	0.12	0.42	0.62	0.64
	4	0.25	0.65	0.73	0.76	0.25	0.60	0.72	0.76
g103	2	0.17	0.30	0.39	0.44	0.15	0.30	0.39	0.43
	4	0.28	0.52	0.63	0.67	0.25	0.45	0.61	0.67

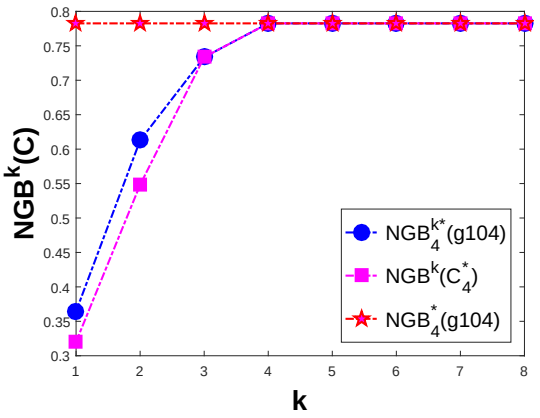


(a) Group size $g = 1$

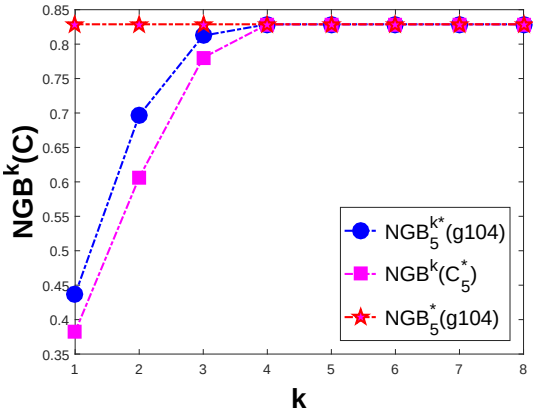
(b) Group size $g = 2$



(c) Group size $g = 3$



(d) Group size $g = 4$



(e) Group size $g = 5$

Figure 6.4: Change in $NGB_g^{k*}(G)$ and $NGB^k(C_g^*)$ for different k and g values for the network g104

6.3 Random Networks

In this section, we consider random graphs to see the effect of density on the optimum k-step normalized GBC value, i.e., $NGB_g^{k*}(G)$. For this purpose, we use the Erdös-Renyi graph model and scale-free networks.

6.3.1 Erdös-Renyi Networks

Erdös-Renyi graph model has two inputs: the number of vertices (n) and probability (p) of adding an edge between vertex $i \in \{1, ..., n\}$ and vertex $j \in \{1, ..., n\}, j \neq i$. As p increases, the graph becomes denser. An Erdös-Renyi graph generated using the inputs n and p is denoted by $ER_{n,p}$. We use different n and p values and for each pair of n and p values, we generate 50 connected Erdös-Renyi graphs $ER_{n,p}$ (if a generated graph is not connected, it is discarded out). Some properties of the generated Erdös-Renyi graphs are given in Table 6.7.

Table 6.7: Some properties of the randomly generated Erdös-Renyi graphs

Graph	Maximum of md	Average of ad	Average of md	Maximum of d_{max}	Average of d_{avg}	Average density
$ER_{20,0.15}$	8.00	2.59	5.38	9	3.17	0.17
$ER_{20,0.20}$	6.00	2.22	4.50	10	4.00	0.21
$ER_{20,0.25}$	5.00	1.99	3.74	11	4.79	0.25
$ER_{20,0.30}$	5.00	1.86	3.36	12	5.54	0.29
$ER_{40,0.06}$	16.00	3.75	8.54	8	2.81	0.07
$ER_{40,0.08}$	9.00	3.12	6.78	10	3.73	0.09
$ER_{40,0.10}$	8.00	2.73	5.56	12	4.07	0.10
$ER_{60,0.06}$	10.00	3.27	6.82	12	3.64	0.06
$ER_{60,0.08}$	7.00	2.75	5.40	13	4.80	0.08
$ER_{60,0.10}$	6.00	2.47	4.72	14	5.93	0.10

* md: maximum distance, ad: average distance, d_{max}: maximum degree, d_{avg}: average degree.

Each value in the table is obtained by considering the values of the 50 graphs gen-

erated using the same inputs and taking the average or the maximum. Note that the average density of the Erdös-Renyi graphs $ER_{n,p}$ is expected to be p. As we have discarded out the disconnected ones, we observe average density values that are slightly bigger than the value of p for some cases in Table 6.7.

For each of the 50 Erdös-Renyi graphs $ER_{n,p}$ generated using the same inputs (and for each fixed k and g values), we compute the optimum k-step normalized GBC value, i.e., $NGB_g^{k*}(ER_{n,p})$, and report the average of the resulting 50 values in Tables 6.8, 6.9, and 6.10. From these tables, we can conclude that the average k-step normalized GBC values are non-increasing as the density, i.e., p, increases. This can be explained as follows. When the density of a graph is small, there are usually a small number of central vertices in the graph and several shortest paths use these central vertices. On the other hand, when the density of a graph is large, the distances between pairs of vertices become very small and several shortest paths use a few number of intermediary vertices if not none. Therefore, in general, we expect a group of central vertices in a dense graph to be on less number of shortest paths when compared to central vertices in a sparse graph.

When we compare the average optimum k-step normalized GBC values for the Erdös-Renyi graph classes $ER_{40,p}$ and $ER_{60,p}$ for the same p and group size g values, we can make the following observations. First, as n increases from 40 to 60, the average optimum k-step normalized GBC values tend to decrease for each k value. This difference is mainly because g many vertices correspond to a higher percentage of all the vertices in the graphs $ER_{40,p}$ than in $ER_{60,p}$. Moreover, as n increases from 40 to 60, the difference between the average optimum k-step normalized GBC values become more noticeable for larger k values (again for fixed p and g).

When we compare the average optimum k-step normalized GBC values for fixed k values when the group size $g = n/20$, we can still see that these values tend to decrease as n increases from 40 to 60.

Table 6.8: Average $NGB_g^{k*}(G)$ for different k and g values on 50 randomly generated Erdös-Renyi graphs of order 20

Graph	**g** \ **k**	**1**	**2**	**3**	**4**	**5**	**6**
$ER_{20,0.15}$	1	0.22	0.33	0.37	0.38	0.38	0.38
	2	0.37	0.55	0.60	0.61	0.61	0.61
	3	0.51	0.71	0.75	0.76	0.76	0.76
	4	0.63	0.82	0.85	0.86	0.86	0.86
	5	0.73	0.90	0.92	0.92	0.92	0.92
	6	0.81	0.94	0.95	0.95	0.95	0.95
$ER_{20,0.20}$	1	0.20	0.29	0.31	0.31	0.31	0.31
	2	0.36	0.50	0.52	0.52	0.52	0.52
	3	0.49	0.65	0.67	0.67	0.67	0.67
	4	0.60	0.76	0.78	0.79	0.79	0.79
	5	0.70	0.85	0.86	0.86	0.86	0.86
	6	0.78	0.90	0.92	0.92	0.92	0.92
$ER_{20,0.25}$	1	0.20	0.27	0.27	0.27	0.27	0.27
	2	0.36	0.46	0.47	0.47	0.47	0.47
	3	0.49	0.61	0.62	0.62	0.62	0.62
	4	0.59	0.72	0.73	0.73	0.73	0.73
	5	0.69	0.81	0.82	0.82	0.82	0.82
	6	0.76	0.88	0.88	0.88	0.88	0.88
$ER_{20,0.30}$	1	0.18	0.23	0.23	0.23	0.23	0.23
	2	0.33	0.41	0.41	0.41	0.41	0.41
	3	0.45	0.55	0.56	0.56	0.56	0.56
	4	0.56	0.67	0.67	0.67	0.67	0.67
	5	0.65	0.76	0.77	0.77	0.77	0.77
	6	0.74	0.84	0.84	0.84	0.84	0.84

Table 6.9: Average $NGB_g^{k*}(G)$ for different k and g values on 50 randomly generated Erdös-Renyi graphs of order 40

Graph	**g** \ **k**	**1**	**2**	**3**	**4**	**5**	**6**
$ER_{40,0.06}$	1	0.12	0.23	0.30	0.33	0.35	0.35
	2	0.22	0.39	0.50	0.54	0.56	0.56
	3	0.31	0.52	0.64	0.68	0.69	0.70
	4	0.39	0.62	0.74	0.78	0.78	0.79
	5	0.46	0.70	0.81	0.84	0.85	0.85
	6	0.52	0.77	0.87	0.89	0.89	0.89
$ER_{40,0.08}$	1	0.11	0.19	0.24	0.25	0.25	0.25
	2	0.21	0.34	0.41	0.43	0.43	0.43
	3	0.29	0.47	0.55	0.57	0.57	0.57
	4	0.37	0.57	0.66	0.68	0.68	0.68
	5	0.44	0.66	0.74	0.76	0.76	0.76
	6	0.50	0.73	0.81	0.82	0.82	0.82
$ER_{40,0.10}$	1	0.11	0.18	0.21	0.21	0.21	0.21
	2	0.21	0.33	0.36	0.37	0.37	0.37
	3	0.29	0.44	0.49	0.50	0.50	0.50
	4	0.37	0.54	0.60	0.61	0.61	0.61
	5	0.43	0.63	0.68	0.69	0.69	0.69
	6	0.50	0.70	0.75	0.76	0.76	0.76

Table 6.10: Average $NGB_g^{k*}(G)$ for different k and g values on 50 randomly generated Erdös-Renyi graphs of order 60

Graph	**g** \ **k**	**1**	**2**	**3**	**4**	**5**	**6**
$ER_{60,0.06}$	1	0.08	0.15	0.19	0.20	0.20	0.20
	2	0.16	0.28	0.33	0.35	0.35	0.35
	3	0.22	0.38	0.45	0.47	0.47	0.47
	4	0.28	0.46	0.55	0.57	0.57	0.57
	5	0.33	0.53	0.63	0.65	0.65	0.65
$ER_{60,0.08}$	1	0.08	0.13	0.14	0.14	0.14	0.14
	2	0.14	0.23	0.26	0.26	0.26	0.26
	3	0.20	0.32	0.35	0.36	0.36	0.36
	4	0.26	0.39	0.44	0.44	0.44	0.44
	5	0.31	0.46	0.52	0.52	0.52	0.52
$ER_{60,0.10}$	1	0.07	0.11	0.12	0.12	0.12	0.12
	2	0.13	0.20	0.22	0.22	0.22	0.22
	3	0.19	0.28	0.30	0.30	0.30	0.30
	4	0.24	0.36	0.38	0.38	0.38	0.38
	5	0.29	0.42	0.45	0.45	0.45	0.45

We also examine absolute and relative differences between $NGB_g^{k*}(G)$ and $NGB^k(C_g^*)$ for different combinations of k and g values for the considered Erdös-Renyi graphs. Results are reported in the Appendix A.2. For any network G and group size g, the difference between $NGB_g^{k*}(G)$ and $NGB^k(C_g^*)$ are generally decreasing in k. Moreover, we can say that as the density of graphs increases, differences decrease in general. Consider the graph $ER_{40,0.06}$. When g=1 and k=1 maximum absolute difference is equal to 4.67 and maximum relative difference is equal to 57.72. Consider the graph $ER_{40,0.10}$. For the same k and g values maximum absolute difference is equal to 1.63 and maximum relative difference is equal to 22.76. Moreover, for the fixed g, as density increases saturation numbers tend to decrease. Consider the graphs $ER_{20,0.15}$, $ER_{20,0.20}$, and $ER_{20,0.25}$. When $g = 3$, maximum saturation num-

bers for the considered 50 sample are $s_3(ER_{20,0.15}) = 5$, $s_3(ER_{20,0.20}) = 3$, and $s_3(ER_{20,0.25}) = 2$.

We next investigate how the running time of Algorithm 1 and the preprocessing step change as n, k, g, and the density change for Erdös-Renyi graphs. For this purpose, we take one graph at random from the 50 Erdös-Renyi graphs generated for each pair of $n \in \{40, 60\}$ and $p \in \{0.06, 0.08, 0.1\}$ values. The matrices d and σ are computed once for each graph and the time to compute them is given in Table 6.11 for each graph in the column titled "Preprocessing Time (in s) to Compute σ and d". Moreover, the matrix PB^k is computed once for each k and each graph. The time to compute PB^k for each k value and each graph is given in Table 6.11 in the column titled "Preprocessing Time (in s) to Compute PB^k". The total time (in s) to compute all $NGB^k(C)$ values for all possible groups C of size g for each graph is given in the following columns.

For group sizes 1 and 3, the computation times are very small. When the group size is 5, Algorithm 1 can compute the optimum normalized GBC value in less than 3 minutes (including the preprocessing time) for each Erdös-Renyi graph and each k value considered. Moreover, once the preprocessing is done, the average time to compute one $NGB^k(C)$ value is between 1.11×10^{-5} and 2.63×10^{-5} for every pair of k and g values for the graphs $ER_{40,p}$ and between 1.08×10^{-5} and 2.64×10^{-5} for every pair of k and g values for the graphs $ER_{60,p}$.

Table 6.11: The preprocessing and total computational time (in s) to compute $NGB^k(C)$ values for all possible groups C of size g on Erdös-Renyi graphs ($ER_{n,p}$)

				Total Time (in s) to Compute $NGB^k(C)$ Values for All Possible Groups C of Size g		
		Prep. time (in s) to	**Prep. time (in s) to**	**Group size g**		
Graph	k	**Compute σ & d**	**Compute PB^k**	**1**	**3**	**5**
$ER_{40,0.06}$	2	0.11	0.00	0.00	0.12	15.30
	4		0.00	0.00	0.12	16.81
	6		0.01	0.00	0.12	18.34
$ER_{40,0.08}$	2	0.09	0.00	0.00	0.12	15.30
	4		0.00	0.00	0.12	17.11
	6		0.00	0.00	0.13	17.28
$ER_{40,0.10}$	2	0.08	0.00	0.00	0.11	15.32
	4		0.00	0.00	0.11	17.12
	6		0.00	0.00	0.11	17.11
$ER_{60,0.06}$	2	0.23	0.01	0.00	0.37	117.33
	4		0.01	0.00	0.38	128.31
	6		0.01	0.00	0.39	130.87
$ER_{60,0.08}$	2	0.18	0.01	0.00	0.37	119.60
	4		0.01	0.00	0.38	134.87
	6		0.02	0.00	0.38	132.94
$ER_{60,0.10}$	2	0.18	0.01	0.00	0.46	133.66
	4		0.01	0.00	0.41	144.03
	6		0.01	0.00	0.40	140.47

* Prep.: Preprocessing

6.3.2 Scale-free Networks

Scale-free random networks are generated by implementing the preferential attachment procedure [73] which has two parameters: n as the number of vertices and e as the number of connections each new vertex will have with the existing vertices. The procedure starts with two vertices connected to each other. In each following step a new vertex is introduced and is attached to e many existing vertices. The probability

that the new vertex is attached to a given existing vertex is taken proportional to the degree of the existing vertex. As e increases, the graph becomes denser. A preferential attachment graph generated using the inputs n and e is denoted by $PA_{n,e}$.

We use different n and e values and for each pair of n and e values, we generate 50 preferential attachment graphs $PA_{n,e}$. Some properties of the generated preferential attachment graphs are given in Table 6.12.

Table 6.12: Some properties of the randomly generated preferential attachment graphs

Graph	Maximum of md	Average of ad	Average of md	Maximum of d_{max}	Average of d_{avg}	Average density
$PA_{20,1}$	8.00	3.11	6.24	13	1.90	0.10
$PA_{20,2}$	5.00	2.14	3.90	14	3.70	0.19
$PA_{20,3}$	4.00	1.84	3.04	15	5.40	0.28
$PA_{40,1}$	11.00	3.75	7.92	26	1.95	0.05
$PA_{40,2}$	5.00	2.55	4.72	23	3.85	0.10
$PA_{40,3}$	4.00	2.18	3.96	26	5.70	0.15
$PA_{60,1}$	13.00	4.18	9.26	26	1.97	0.03
$PA_{60,2}$	6.00	2.76	5.02	28	3.90	0.07
$PA_{60,3}$	4.00	2.36	4.00	30	5.80	0.10

* md: maximum distance, ad: average distance, d_{max}: maximum degree, d_{avg}: average degree.

Average of the optimum k-step normalized GBC values ($NGB_g^{k*}(G)$) for 50 randomly generated preferential attachment graphs $PA_{n,e}$ are reported in Tables 6.13 and 6.14 for each pair of n and e values. From these tables, we can conclude that the average optimum k-step normalized GBC values decrease as the density increases, i.e., as e increases. When we compare the average $NGB_g^{k*}(G)$ values for the network classes $PA_{20,e}$, $PA_{40,e}$, and $PA_{60,e}$ for the same e and group size g values, we can see that as n increases, the average optimum k-step normalized GBC values decrease for each k value. This can be explained by the fact that for a fixed e, as n increases, the graphs become sparser and as a result the optimum k-step normalized GBC values tend to decrease.

Table 6.13: Average $NGB_g^{k*}(G)$ for different k and g values on 50 randomly generated scale-free networks of order 20 and 40

Graph	**g** \ **k**	**1**	**2**	**3**	**Graph**	**g** \ **k**	**1**	**2**	**3**
$PA_{20,1}$	1	0.41	0.67	0.78	$PA_{40,1}$	1	0.31	0.58	0.73
	2	0.62	0.87	0.93		2	0.47	0.78	0.89
	3	0.76	0.95	0.97		3	0.59	0.89	0.95
	4	0.85	0.98	0.98		4	0.67	0.94	0.97
	5	0.92	0.99	0.99		5	0.74	0.97	0.98
	6	0.96	1.00	1.00		6	0.79	0.98	0.99
$PA_{20,2}$	1	0.33	0.43	0.43	$PA_{40,2}$	1	0.24	0.36	0.38
	2	0.53	0.67	0.68		2	0.40	0.58	0.60
	3	0.67	0.81	0.82		3	0.51	0.71	0.73
	4	0.77	0.89	0.90		4	0.60	0.80	0.82
	5	0.84	0.94	0.94		5	0.66	0.86	0.87
	6	0.90	0.97	0.97		6	0.72	0.89	0.91
$PA_{20,3}$	1	0.25	0.29	0.29	$PA_{40,3}$	1	0.20	0.26	0.26
	2	0.44	0.50	0.50		2	0.35	0.44	0.44
	3	0.58	0.66	0.66		3	0.46	0.57	0.58
	4	0.69	0.77	0.77		4	0.54	0.67	0.67
	5	0.77	0.85	0.85		5	0.61	0.74	0.75
	6	0.84	0.91	0.91		6	0.67	0.80	0.80

Preferential attachment graphs contain a few vertices that are highly connected to others [73]. Therefore, the distance between any two randomly chosen vertices tends to be small. This makes the saturation numbers of the preferential attachment graphs small as well. For example, the saturation number of each randomly generated graph in the graph class $PA_{20,3}$ is at most 3.

The average density of the graphs $PA_{40,2}$ is the same as that of the graphs $ER_{40,0.10}$. When we investigate the average optimum k-step normalized GBC values for both graph classes, we can see that the values are much larger for $PA_{40,2}$. Moreover,

we take one random graph for $ER_{40,0.10}$ and $PA_{40,2}$ and examine the distribution of the k-step normalized GBC values of all subsets of V for different k and g values. For each $k \in \{1, 2, 3\}$, we compute $NGB^k(C)$ for every subset C of V of size $g \in \{1, 2, 3, 4, 5\}$ and make a histogram of the $NGB^k(C)$ values (see Figures A.1 and A.2). It can be seen that the distribution of the $NGB^k(C)$ values is right-skewed in scale-free network. However, the distribution of the $NGB^k(C)$ values is right-skewed for $k = 1$ and $g \in \{1, 2\}$, but normally distributed for $k \in \{2, 3\}$ and $g \in \{4, 5\}$. This can be attributed to the existence of highly connected vertices in scale free graphs. These central vertices can control a higher proportion of the information flow in preferential attachment graphs.

Table 6.14: Average $NGB_g^{k*}(G)$ for different k and g values on 50 randomly generated scale-free networks of order 60

Graph	**g** \ **k**	**1**	**2**	**3**	**4**	**5**
$PA_{60,1}$	1	0.25	0.52	0.69	0.77	0.80
	2	0.39	0.71	0.86	0.90	0.91
	3	0.49	0.82	0.93	0.95	0.95
	4	0.56	0.88	0.96	0.97	0.97
	5	0.63	0.92	0.97	0.98	0.98
$PA_{60,2}$	1	0.20	0.33	0.35	0.35	0.35
	2	0.34	0.53	0.56	0.56	0.56
	3	0.44	0.65	0.68	0.68	0.68
	4	0.51	0.74	0.77	0.77	0.77
	5	0.58	0.80	0.83	0.83	0.83
$PA_{60,3}$	1	0.16	0.23	0.23	0.23	0.23
	2	0.28	0.39	0.39	0.39	0.39
	3	0.38	0.51	0.52	0.52	0.52
	4	0.46	0.61	0.62	0.62	0.62
	5	0.53	0.68	0.69	0.69	0.69

We also investigate how the running time of Algorithm 1 and the preprocessing step

change as n, k, g, and the density change for preferential attachment graphs. For this purpose, we take one graph at random from the 50 preferential attachment graphs generated for each pair of $n \in \{40, 60\}$ and $e \in \{1, 2, 3\}$ values. The matrices d and σ are computed once for each graph and the time to compute them is given in Table 6.15 for each graph in the column titled "Preprocessing Time (in s) to Compute σ and d". Moreover, the matrix PB^k is computed once for each k and each graph. The time to compute PB^k for each k value and each graph is given in Table 6.15 in the column titled "Preprocessing Time (in s) to Compute PB^k". The total time (in s) to compute all $NGB^k(C)$ values for all possible groups C of size g for each graph is given in the following columns.

Table 6.15: The preprocessing and total computational time (in s) to compute $NGB^k(C)$ values for all possible groups C of size g on preferential attachment graphs ($PA_{n,e}$)

				Total Time (in s) to Compute $NGB^k(C)$ Values for All Possible Groups C of Size g		
		Prep. time (in s) to	**Prep. time (in s) to**	**Group size g**		
Graph	k	**Compute σ & d**	**Compute PB^k**	**1**	**3**	**5**
$PA_{40,1}$	1	0.07	0.00	0.00	0.06	7.61
	2		0.00	0.00	0.06	7.70
	3		0.00	0.00	0.06	7.86
$PA_{40,2}$	1	0.07	0.00	0.00	0.06	8.11
	2		0.00	0.00	0.06	7.83
	3		0.00	0.00	0.06	7.97
$PA_{40,3}$	1	0.07	0.00	0.00	0.06	7.67
	2		0.00	0.00	0.06	7.92
	3		0.00	0.00	0.06	8.03
$PA_{60,1}$	1	0.14	0.00	0.00	0.22	62.78
	2		0.00	0.00	0.22	63.34
	3		0.00	0.00	0.22	64.28
$PA_{60,2}$	1	0.15	0.00	0.00	0.22	63.05
	2		0.00	0.00	0.22	64.11
	3		0.00	0.00	0.22	65.82
$PA_{60,3}$	1	0.15	0.00	0.00	0.22	63.34
	2		0.00	0.00	0.22	65.51
	3		0.00	0.00	0.22	66.36

* Prep.: Preprocessing

For group sizes 1 and 3, the computation times are very small. When the group size is 5, Algorithm 1 can compute the optimum normalized GBC value in about 2 minutes (including the preprocessing time) for each preferential attachment graph and each k value considered. Moreover, once the preprocessing is done, the average time to compute one $NGB^k(C)$ value is between 9.64×10^{-8} and 1.3×10^{-5} for every pair of k and g values for the graphs $PA_{40,e}$ and between 9.58×10^{-8} and 0.83×10^{-5} for

every pair of k and g values for the graphs $PA_{60,e}$.

6.4 Large Scale Networks

Up to this point, we have tested Algorithm 1 on small size networks and observed that when the group size g is small, we can find the group of size g having the highest normalized k-step GBC value by enumerating all possible groups of size g and computing the normalized k-step GBC for each one. When the size of the network is large, it will be impractical if not impossible to enumerate all possible groups of a given size. Still, Algorithm 1 can be useful in some cases for large size networks.

Consider, for instance, a real-life Facebook network [74] consisting 4,039 vertices and 88,234 edges. Here, the nodes represent Facebook users and the existence of an edge between two nodes shows that the corresponding users are friends. Assume that a data analyst is interested in finding the group of size 100 having the highest normalized 8-step GBC value on this Facebook network among a predetermined 1 million candidate groups. Then using Algorithm 1, these 1 million groups can be evaluated and the group having the highest 8-step normalized GBC value can be returned to the data analyst. We will talk more about cases where such candidate groups may appear later. To see how Algorithm 1 performs on this large size Facebook network, we generate 10,000 random subsets for different group sizes.

For this network, the sigma and distance matrices are computed in about 42 minutes. The matrix PB^k is computed once for each k value. The time to compute PB^k for each k value is given in Table 6.16 in the column titled "Prep. time (in s) to Compute PB^k". The total time (in s) to compute all $NGB^k(C)$ values for the randomly generated 10,000 groups C of size g is given in the following columns. We can see from the table that as k and g increase, the computational times tend to increase as well. Going back to evaluating the normalized 8-step GBC of the 1 million groups provided by the data analyst, we can see that Algorithm 1 can do this evaluation in about 22 hours ($42 \times 60 + 4679.23$ seconds preprocessing time $+ 715.88 \times 100$ seconds to compute the 8-step GBC values of the 1 million groups) assuming that the time to evaluate each normalized 8-step GBC remains about the same.

Table 6.16: The preprocessing and total computational time (in s) to compute $NGB^k(C)$ values for randomly selected 10000 groups C of size g on the Facebook network

		Total Time (in s) to Compute $NGB^k(C)$ Values for All Possible Groups C of Size g									
	Prep. Time (in s) to	**Group Size g**									
k	**Compute PB^k**	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
1	3278.94	1.11	4.79	13.98	31.67	59.50	100.92	157.96	235.03	329.93	453.30
2	3359.18	1.02	5.37	15.79	35.66	67.18	114.25	179.06	266.40	374.11	510.57
3	3657.87	1.11	6.00	17.99	40.88	77.40	131.49	206.45	306.76	432.04	593.07
4	4227.36	1.19	6.73	20.47	46.78	88.72	150.50	236.90	351.77	496.80	678.63
5	4569.89	1.21	6.97	21.28	48.68	91.98	175.10	253.28	367.41	514.96	703.44
6	4634.01	1.22	7.05	21.50	49.05	93.15	158.29	248.62	371.14	521.48	712.72
7	4670.12	1.22	7.06	21.58	49.34	93.68	159.16	249.78	371.49	523.30	715.73
8	4679.23	1.22	7.06	21.58	49.32	93.76	159.33	250.14	371.50	523.81	715.88

* Prep.: Preprocessing

In some applications, certain vertices of a network may be distinguished and one may want to find a group of central vertices that are chosen among these distinguished vertices. Consider, for example, a road network where one is allowed to place road side equipments that can monitor the network at only a preselected subset of the vertices of the network. In this case, one may be interested in finding a subset of these preselected vertices having the highest normalized k-step GBC value. In such a case, the number of candidate groups may substantially be smaller than the number of all subsets of all the vertices of a given size.

For the Facebook network, we preselect 20 vertices uniformly at random and we want to find a subset of these preselected vertices having the highest normalized k-step GBC value for different k and g values. The matrices σ, d, and $PB^k, k \in \{1, 2, \ldots, 8\}$ are already computed before, so we do not report the preprocessing times once again here. The total time (in s) to compute all $NGB^k(C)$ values for subsets C of the 20 preselected vertices of size g is given in Table 6.17 for different k and g values. For example, once the preprocessing is done, a group of 10 vertices chosen among the preselected ones having the highest normalized 8-step GBC can be

found in less than 21 seconds. The average time to compute each $NGB^k(C)$ value ranges between 8.77×10^{-6} and 1.10×10^{-4} seconds for every pair of k and g values considered in Table 6.17 for this network.

Table 6.17: The preprocessing and total computational time (in s) to compute $NGB^k(C)$ values for all possible subsets C of the 20 preselected vertices of size g on the Facebook network

	Total Time (in s) to Compute $NGB^k(C)$ Values for All Possible Groups C of Size g Consisting of Preselected Vertices									
	Group Size g									
k	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	0.00	0.00	0.01	0.08	0.34	1.14	3.05	6.36	10.87	14.90
2	0.00	0.00	0.01	0.08	0.36	1.19	3.20	6.82	11.68	16.14
3	0.00	0.00	0.01	0.08	0.37	1.29	3.52	7.90	13.28	18.44
4	0.00	0.00	0.01	0.09	0.39	1.37	3.75	8.04	14.16	19.95
5	0.00	0.00	0.01	0.09	0.40	1.37	3.79	8.24	14.38	20.26
6	0.00	0.00	0.02	0.09	0.40	1.38	3.79	8.15	14.36	20.27
7	0.00	0.00	0.01	0.09	0.40	1.38	3.79	8.18	14.49	20.26
8	0.00	0.00	0.01	0.09	0.40	1.46	4.00	8.25	14.49	20.23

CHAPTER 7

ALTERNATIVE SOLUTION METHODS

Once the preprocessing step is performed, the algorithm proposed by Puzis et al. and Algorithm 1 can evaluate the GBC and k-step GBC, respectively, of several groups of size g, successively very fast. Therefore, in this study we consider it as the basic algorithm to compute k-step BC and k-step GBC. However, when g is not small, the problem to compute the group with the highest GBC or k-step GBC from all possible subsets becomes challenging to solve by the k-step GBC.

Given a ground set X, a weight $w(x)$ for each $x \in X$, a collection $S = \{S_1, S_2, \ldots, S_N\}$ of subsets of X, and a positive integer g, the problem of selecting at most g subsets from S to maximize the total weight of the elements covered by the selected subsets is called as the weighted maximum coverage problem. This problem is NP-hard, and it can be approximated using a greedy algorithm within a ratio of $1 - 1/e = 0.63$ [75]. In other words, the algorithm finds a set whose objective function value is at least "$(1-1/e) \times$ optimal objective function value". Our problem is also a maximum coverage problem with X being the set of all shortest paths, each subset in S consisting of the set of shortest paths passing through a specific node, g being the group size, and the weight of a shortest path joining s to t being $1/\sigma_{st}$. Inspired by maximum coverage problem Dolev et al. [10] show that the problem of finding a set of vertices of size g that maximizes the probability of detecting information sent through a randomly chosen shortest path between randomly chosen vertices is NP-hard. In [8], Puzis et al. propose a greedy heuristic to compute a set of vertices with high GBC value. This algorithm is the same as the approximation algorithm proposed for the weighted maximum coverage problem in the literature. In this chapter, we propose a heuristic algorithm, which is the approximation algorithm proposed for the weighted

maximum coverage problem, to find a group of vertices of size g whose k-step GBC value is high.

Moreover, recently, Veremyev et al. propose a mixed integer programming formulation to find a group of vertices of size g that has the highest GBC value. In this chapter, we also introduce a mixed integer programming model to compute the group of size g having the largest k-step GBC value which is a modification of the model proposed by Veremyev et al. in [9].

7.1 Modeling by Mixed Integer Programs

Now, we introduce a mixed integer programming (MIP) model to compute the group of size g having the largest k-step GBC value. This formulation is a modification of the one proposed by Veremyev et al. [9] for the computation of the optimum group of vertices. We first introduce related parameters and decision variables, afterwards we explain the MIP model.

The notation used for the mathematical model are as follows.

Parameters:

P_{st} : the set of all shortest paths joining s to t. The cardinalities of all members of P_{st} are identical, e.g., $|P_{r_1}| = |P_{r_2}|$ where r is a possible path joining s to t with $r \in \{1, 2, \ldots, \sigma_{st}\}$. Moreover, if s and t are adjacent, the shortest path joining s to t is unique, so $|P_{st}| = 1$. The algorithm that is used to compute all paths between any pair of vertices is available in Appendix A.2

$P_r \in P_{st}$: represents an ordered list of vertices that lie on the r^{th} shortest path joining s to t, including starting and ending vertices (s, t).

σ_{st} : total number of shortest paths joining s to t.

$d(s, t)$: length of the shortest path joining s to t.

g : group size.

Decision variables:

$y_{st}^r : y_{st}^r$ is a binary variable where $\{s,t\} \in V$ and $r \in \{1,2,\ldots,\sigma_{st}\}$. y_{st}^r is equal to 1 if the shortest path $P_r \in P_{st}$ joining s to t passes through at least one vertex in C in the first k steps. In other words, y_{st}^r is equal to 1 if and only if there is a vertex q satisfying all conditions $q \in C$ and $q \in P_r$ and $d(s,q) \leq k$.

$x_s : x_s \in V$ is a binary variable, such that $x_s = 1$ if vertex s is a member of selected group C, and $x_s = 0$ otherwise.

After the relevant notation is introduced, now we define mathematical model.

$$\text{maximize} \quad \sum_{s,t \in V, s \neq t} \frac{\sum_{P_r \in P_{st}} y_{st}^r}{\sigma_{st}} \tag{7.1}$$

$$\text{subject to} \quad y_{st}^r \leq \sum_{\substack{q:\ q \in P_r \\ d(s,q) \leq k}} x_q, \quad \forall\, s,t \in V,\ s \neq t,\ \forall\, P_r \in P_{st}, \tag{7.2}$$

$$\sum_{s \in V} x_s = g, \tag{7.3}$$

$$x_s \in \{0,1\}, \qquad 0 \leq y_{st}^r \leq 1\ \forall\, s,t \in V,\ s \neq t,\ \forall\, P_r \in P_{st}. \tag{7.4}$$

Objective function in Equation 7.1 aims to maximize the total fraction of shortest paths that the information is obtained by at least one group member within the first k steps of the start of the communication between all pairs of distinct vertices. Constraint 7.2 ensures that y_{st}^r is equal to 1 in an optimal solution if at least one vertex of the related shortest path P_r is covered by C in the first k steps, and it is equal to 0 otherwise. Constraint 7.3 guarantees that cardinality of the selected group of vertices doesn't exceed defined group size g. Finally, constraint 7.4 defines bounding limitations and binary restrictions for variables x_s and y_{st}^r . Note that binary restrictions for y_{st}^r are relaxed, as it is a maximization problem and existence of constraint 7.2.

Considered MIP model consist of n binary variables, $\sum_{s,t \in V, s \neq t} \sigma_{st}$ many continuous variables, and $\sum_{s,t \in V, s \neq t} \sigma_{st} + 1$ many constraints.

7.2 An Approximation Algorithm to Compute the k-step Group Betweenness Centrality

Now, we describe an algorithm to find a group whose k-step GBC value is high for a given group size g which is a modification of the algorithm proposed by Puzis et al. in [7].

Algorithm 2 determine a group of vertices of size g iteratively. The algorithm starts choosing the vertex v with the highest k-step BC value and at each iteration it adds a new vertex $v \in V \setminus \{M\}$ according to its contribution to k-step GBC value of existing group members. This algorithm is a $1 - 1/e$ approximation algorithm [10] for the problem of finding a gorup of vertices with size g with high k-step GBC value. The steps of the Algorithm 2 are given in Algorithm 2.

The inputs of the algorithm are a graph $G = (V, E)$, group size g, and a positive integer k, and the outputs are the group $M \subseteq V$ of size g and $GB^k(M)$ value. Algorithm 2 starts with a preprocessing step just like Algorithm 1. During the algorithm, the contributions of selected elements from V on $GB^k(M)$ are considered one by one between lines 8 and 28. A set M is defined in line 4 which is initially an empty set, and the algorithm ends when M has g entry. Once their contributions to $GB^k(M)$ are considered, the elements of V are added one by one to M during the algorithm. We define two $n \times n$ matrices $\sigma^M = [\sigma^M_{xy}]_{x,y \in V}$ and $PB^k_M = [PB^k_M(x, y)]_{x,y \in V}$. Initially in lines 5 and 6, we have that $\sigma^M_{xy} = \sigma_{xy}$ and $PB^k_M(x, y) = PB^k(x, y), \forall x, y \in V$ as M is the empty set for now. Moreover $GB^k(M)$ is initialized to zero in line 7 as the contribution of any element of V is not yet considered. In each iteration within the for loop between lines 8 and 28, the vertex which have highest $PB^k_M(v, v)$ value, say v^*, is chosen from candidate vertices $V \setminus M$ and, its contribution to $GB^k(M)$ is considered. This is done in line 10 by the equation $GB^k(M \cup \{v^*\}) = GB^k(M) + PB^k_M(v^*, v^*)$. As M will be updated to $M \cup \{v^*\}$, to be able to determine the contribution of the following elements of V to $GB^k(M)$, the matrix PB^k_M has to be updated. For this purpose, we first update the matrix σ^M between lines 12 and 14. For any $x, y \in V$, σ^M_{xy} is subject to change when v^* lies on at least one shortest path joining x to y. Between lines 15 and 25, the matrix PB^k_M is updated. The procedures to update σ^M and PB^k_M are just like in Algorithm 1. Finally, in line 27, M is updated as $M \cup \{v^*\}$.

The preprocessing step of Algorithm 2 takes $O(n^3)$ time. Algorithm 2 spends $O(n^2)$ time within the for loop between lines 11 and 26. This loop repeats g times iteratively, so the total computational time is $O(n^3 + gn^2)$.

Algorithm 2 k-step GBC - Heuristic

1: Inputs: $G = (V, E)$, g, and a positive integer k
2: Outputs: $M \subseteq V$ and $GB^k(M)$
3: Preprocessing: Compute σ, d, and PB^k.
4: $M \leftarrow \{\}$
5: $\sigma^M_{xy} \leftarrow \sigma_{xy}, \forall x, y \in V$
6: $PB^k_M(x, y) \leftarrow PB^k(x, y), \forall x, y \in V$
7: $GB^k(M) \leftarrow 0$
8: **for** $i = 1$ to g **do**
9: $v^* = argmax_v PB^k_M(v, v)$
10: $GB^k(M) \leftarrow GB^k(M) + PB^k_M(v^*, v^*)$
11: **for** $\forall x, y \in V$ **do**
12: **if** $d(x, y) = d(x, v^*) + d(v^*, y)$ **then**
13: $\sigma^{M\cup\{v^*\}}_{xy} \leftarrow \sigma^M_{xy} - \sigma^M_{xv^*}\sigma^M_{v^*y}$
14: **end if**
15: **if** $x = y \neq v^*$ **then**
16: $PB^k_{M\cup\{v^*\}}(x, x) \leftarrow PB^k_M(x, x) - PB^k_M(v^*, x) - PB^k_M(x, v^*)$
17: **else if** $d(v^*, y) = d(v^*, x) + d(x, y)$ **then**
18: $PB^k_{M\cup\{v^*\}}(x, y) \leftarrow PB^k_M(x, y) - \frac{\sigma^M_{v^*x}\sigma^M_{xy}PB^k_M(v^*,y)}{\sigma^M_{v^*y}}$
19: **else if** $d(x, y) = d(x, v^*) + d(v^*, y)$ **then**
20: $PB^k_{M\cup\{v^*\}}(x, y) \leftarrow PB^k_M(x, y) - \frac{\sigma^M_{xv^*}\sigma^M_{v^*y}PB^k_M(x,y)}{\sigma^M_{xy}}$
21: **else if** $d(x, v^*) = d(x, y) + d(y, v^*)$ **then**
22: $PB^k_{M\cup\{v^*\}}(x, y) \leftarrow PB^k_M(x, y) - \frac{\sigma^M_{xy}\sigma^M_{yv^*}PB^k_M(x,v^*)}{\sigma^M_{xv^*}}$
23: **else**
24: $PB^k_{M\cup\{v^*\}}(x, y) \leftarrow PB^k_M(x, y)$
25: **end if**
26: **end for**
27: $M \leftarrow M \cup \{v^*\}$
28: **end for**

7.3 Computational Experiments

In this section, we perform several computational experiments on real and randomly generated networks to

- understand the change in running time of Algorithm 2 and MIP model as k and g change.
- understand success rate of Algorithm 2 as k, g, and the density change.

7.3.1 Real-life Networks

The first real network is rt-retweet network where vertices represent twitter users, and edges represent hashtags about social and political issues. Secondly, we use jazz-musicians network. Here, the vertices represent jazz musicians, and the existence of an edge between two vertices shows that the corresponding users have recorded in the same band. The third real-life network is ca-netscience where each vertex represents a scientist, and edges show that there is co-authorship between scientists [71].

Some properties of the considered real-life networks are given in Table 7.1.

Table 7.1: Some properties of the considered real-life networks.

Network	$\|V\|$	$\|E\|$	**md**	**ad**	d_{max}	d_{avg}	**Density**
rt-retweet	96	117	10	4.31	17	2.44	0.03
jazz-musicians	196	2742	6	2.24	100	27.70	0.14
ca-netscience	379	914	17	6.04	34	4.82	0.01

* md: maximum distance, ad: average distance, d_{max}: maximum degree, d_{avg}: average degree.

Now, we investigate how the running times of Algorithm 2 and MIP formulation change as n, k, and g change for real-life networks. We report total time (in s) to compute the group of size g having the largest k-step GBC value in the columns titled "MIP Solution" in Table 7.2. Moreover, we report total time (in s) to find a group

whose k-step GBC value is high for a given group of size g in the columns titled "Heuristic Solution" in Table 7.2

Table 7.2: Total time (in s) to compute the group of size g having the largest k-step GBC value and total time (in s) to find a group whose k-step GBC value is high for a given group of size g on real-life networks

		MIP Solution				**Heuristic Solution**			
Network	**k** \ **g**	**5**	**10**	**15**	**20**	**5**	**10**	**15**	**20**
rt-retweet	2	2.81	1.23	0.66	0.25	0.01	0.01	0.02	0.02
	4	1.60	0.42	0.34	0.38	0.01	0.01	0.02	0.02
	6	1.56	0.44	0.34	0.36	0.01	0.01	0.01	0.02
	8	1.70	0.44	0.34	0.31	0.01	0.01	0.01	0.02
jazz-musicians	2	969.55	2566.59	2037.16	3286.91	0.03	0.05	0.08	0.10
	4	1376.98	2124.00	2757.11	14855.00	0.03	0.06	0.09	0.12
	6	1366.92	1976.16	3448.88	24626.89	0.03	0.06	0.08	0.11
	8	1379.42	1854.92	3512.16	24649.32	0.03	0.07	0.08	0.11
ca-netscience	2	1672.77	1644.45	1407.30	1016.20	0.12	0.19	0.27	0.35
	4	2203.55	768.23	656.47	319.47	0.11	0.19	0.28	0.34
	6	1474.31	611.77	300.86	336.05	0.12	0.21	0.28	0.35
	8	1709.11	612.50	330.20	302.56	0.12	0.22	0.31	0.38

For the rt-retweet network which has 96 vertices, computation times are very small for both solution methods. MIP model can compute the optimum normalized k-step GBC value in less than 3 seconds for each of the k and g values. Also, Algorithm 2 can compute the group whose k-step GBC value is high in less than 0.03 seconds for each of the k and g values. As g and n increase, total time (in s) to compute heuristic solution increases as expected. Also, as n and graph density increase running time of MIP model increases dramatically. Consider the jazz-musicians network which has 196 vertices. When $k = 6$ and $g = 20$, exact solution is found in 24626.89 seconds, but heuristic solution takes just 0.11 seconds. Moreover, even when n increases from 96 to 379, Algorithm 2 can do this evaluation in less than 0.4 seconds for all k and g values.

Remember that Algorithm 2 returns a subset M whose k-step GBC value is high,

but it doesn't guarantee optimum result. Therefore, it is possible to see an optimality GAP between $NGB_g^{k*}(G)$ value and $NGB_g^k(M)$ value where

$$\text{Optimality GAP} = \frac{NGB_g^{k*}(G) - NGB_g^k(M)}{NGB_g^{k*}(G)}. \tag{7.5}$$

Now, we investigate how optimality GAP changes as n, k, and g change for real-life networks. We report $NGB_g^{k*}(G)$, $NGB_g^k(M)$, and Optimality GAP values for different k and g values in Table 7.3. Observe that gaps are generally small for different networks, and k and g combinations. Also as g increases gaps tend to be increase. When $g = 5$ the maximum gap is equal to 0.03 and mean gap is equal to 0.01, and when $g = 20$ the maximum gap is equal to 0.08 and mean gap is equal to 0.03.

Table 7.3: Optimum k-step GBC value ($NGB_g^{k*}(G)$), k-step GBC value of the group of vertices M returned by Algorithm 2 ($NGB_g^k(M)$) and Optimality GAP for different k and g values on real-life networks

		$NGB_g^{k*}(G)$				$NGB_g^k(M)$				**Optimality GAP**			
Network	**g / k**	**5**	**10**	**15**	**20**	**5**	**10**	**15**	**20**	**5**	**10**	**15**	**20**
rt-retweet	1	0.42	0.62	0.76	0.87	0.42	0.61	0.73	0.80	0.00	0.02	0.04	0.08
	2	0.73	0.92	0.98	1.00	0.73	0.89	0.96	0.98	0.00	0.03	0.02	0.02
	3	0.86	0.98	0.99	1.00	0.86	0.98	0.99	0.99	0.00	0.00	0.00	0.01
	4	0.90	0.98	0.99	1.00	0.90	0.98	0.99	0.99	0.00	0.00	0.00	0.01
	5	0.91	0.98	0.99	1.00	0.91	0.98	0.99	0.99	0.00	0.00	0.00	0.01
jazz-musicians	1	0.30	0.42	0.51	0.58	0.29	0.41	0.50	0.57	0.03	0.02	0.02	0.02
	2	0.38	0.53	0.63	0.71	0.37	0.52	0.62	0.69	0.03	0.02	0.02	0.03
	3	0.39	0.54	0.64	0.72	0.38	0.53	0.63	0.70	0.03	0.02	0.02	0.03
	4	0.39	0.54	0.64	0.72	0.39	0.53	0.63	0.70	0.00	0.02	0.02	0.03
	5	0.39	0.54	0.64	0.72	0.39	0.53	0.63	0.70	0.00	0.02	0.02	0.03
ca-netscience	1	0.24	0.37	0.48	0.56	0.24	0.37	0.46	0.53	0.00	0.00	0.04	0.05
	2	0.52	0.72	0.84	0.90	0.52	0.72	0.83	0.87	0.00	0.00	0.01	0.03
	3	0.69	0.87	0.94	0.96	0.69	0.87	0.94	0.96	0.00	0.00	0.00	0.00
	4	0.78	0.92	0.96	0.97	0.76	0.90	0.94	0.95	0.03	0.02	0.02	0.02
	5	0.82	0.93	0.96	0.97	0.81	0.92	0.94	0.95	0.01	0.01	0.02	0.02

7.3.2 Random Networks

Now, we investigate how the running times of Algorithm 2 and MIP formulation change as n, k, g, and the density change on randomly generated Erdös-Renyi graphs. For this purpose, we generate connected Erdös-Renyi graphs $ER_{n,p}$ for each pair of n and p values, where $n \in \{100, 150, 200\}$ and $p \in \{0.03, 0.1\}$. Some properties of the generated Erdös-Renyi graphs are given in Table 7.4

Table 7.4: Some properties the randomly generated Erdös-Renyi graphs

Network	$\|V\|$	$\|E\|$	**md**	**ad**	d_{max}	d_{avg}
$ER_{100,03}$	100	173	10	3.96	7	3.46
$ER_{100,10}$	100	483	4	2.26	17	9.66
$ER_{150,03}$	150	339	7	3.45	10	4.52
$ER_{150,10}$	150	1070	4	2.14	25	14.27
$ER_{200,03}$	200	585	6	3.19	13	5.85
$ER_{200,10}$	200	2029	3	2.01	32	20.29

* md: maximum distance, ad: average distance, d_{max}: maximum degree, d_{avg}: average degree.

We report total time (in s) to compute the group of size g having the largest k-step GBC value in the columns titled "MIP Solution" in Table 7.5. Moreover, we report total time (in s) to find a group whose k-step GBC value is high for a given group of size g in the columns titled "Heuristic Solution" in Table 7.5.

We can see that running times of both algorithms tend to increase as n or g increase. Observe that while running time of Algorithm 2 doesn't depend on density p, running time of MIP model increases as p increases. Consider the graph $ER_{150,0.03}$. When $k = 2$ and $g = 10$, total time to compute $NGB_g^{k*}(G)$ takes 50.78 seconds, and for the graph $ER_{150,0.10}$ total time to compute $NGB_g^{k*}(G)$ takes 375.33 seconds for the same k and g values. Moreover, when g and n increase, the difference is generally much more pronounced. Consider the graph $ER_{200,0.03}$. When $k = 4$ and $g = 20$, total time to compute $NGB_g^{k*}(G)$ takes 2191.55 seconds, and for the graph $ER_{200,0.10}$ total time to compute $NGB_g^{k*}(G)$ takes 107280.22 seconds for the same k and g values.

Table 7.5: Total time (in s) to compute the group of size g having the largest k-step GBC value and total time (in s) to find a group whose k-step GBC value is high for a given group of size g on randomly generated Erdös-Renyi graphs

		MIP Solution				**Heuristic Solution**			
Graph	**g** / **k**	**5**	**10**	**15**	**20**	**5**	**10**	**15**	**20**
$ER_{100,0.03}$	2	5.07	8.41	8.45	5.69	0.01	0.01	0.02	0.03
	4	13.77	20.94	26.30	21.36	0.01	0.01	0.01	0.02
	6	16.13	22.08	27.30	24.63	0.01	0.01	0.01	0.02
$ER_{100,0.10}$	2	30.44	46.20	96.63	222.25	0.01	0.01	0.02	0.03
	4	37.02	97.78	153.58	505.42	0.01	0.02	0.03	0.02
$ER_{150,0.03}$	2	32.23	50.78	65.81	116.78	0.02	0.03	0.04	0.04
	4	82.09	180.50	272.84	1530.13	0.02	0.03	0.05	0.05
	6	74.50	154.19	324.11	1742.92	0.01	0.03	0.04	0.05
$ER_{150,0.10}$	2	192.14	375.33	854.24	4381.28	0.02	0.04	0.05	0.05
	4	459.77	614.03	13025.70	26846.08	0.01	0.03	0.05	0.05
$ER_{200,0.03}$	2	163.52	187.80	258.48	289.42	0.02	0.03	0.05	0.07
	4	358.70	555.49	671.00	2191.55	0.03	0.06	0.08	0.09
	6	364.08	751.89	674.44	2828.78	0.03	0.06	0.08	0.09
$ER_{200,0.10}$	2	1359.25	1730.58	3580.89	46734.77	0.02	0.04	0.05	0.07
	4	1165.81	2816.48	6332.36	107280.22	0.02	0.04	0.05	0.07

Now, we investigate how optimality GAP changes as n, k, g, and the density change on randomly generated Erdös-Renyi graphs. We report $NGB_g^{k*}(G)$, $NGB_g^k(M)$, and optimality GAP values in Table 7.6. We can see that gaps are smaller than real-life networks. When $g = 5$, Algorithm 2 gives optimum result for all k and g combinations. When $g = 10$, Algorithm 2 gives optimum result except of graph $ER_{200,0.1}$. When $g = 15$ and $g = 20$ maximum gaps are 0.04 and 0.02, respectively. Observe that as density (p) increases, gaps tend to decrease generally. Consider graphs $ER_{100,0.1}$ and $ER_{150,0.1}$. Algorithm 2 gives optimum result for all k and g combinations.

Table 7.6: Optimum k-step GBC value ($NGB_g^{k*}(G)$), k-step GBC value of the group of vertices M returned by Algorithm 2 ($NGB_g^k(M)$) and Optimality GAP for different k and g values on Erdös-Renyi graphs.

		$NGB_g^{k*}(G)$				$NGB_g^k(M)$				**Optimality GAP**			
Graph	**g** / **k**	**5**	**10**	**15**	**20**	**5**	**10**	**15**	**20**	**5**	**10**	**15**	**20**
$ER_{100,0.03}$	1	0.20	0.36	0.50	0.62	0.20	0.36	0.50	0.62	0.00	0.00	0.00	0.00
	2	0.36	0.58	0.74	0.86	0.36	0.58	0.73	0.85	0.00	0.00	0.01	0.01
	3	0.47	0.71	0.84	0.93	0.47	0.71	0.84	0.91	0.00	0.00	0.00	0.02
	4	0.51	0.76	0.88	0.95	0.51	0.76	0.87	0.93	0.00	0.00	0.01	0.02
	5	0.52	0.77	0.89	0.95	0.52	0.77	0.88	0.94	0.00	0.00	0.01	0.01
$ER_{100,0.10}$	1	0.17	0.30	0.43	0.53	0.17	0.30	0.43	0.53	0.00	0.00	0.00	0.00
	2	0.24	0.42	0.57	0.69	0.24	0.42	0.57	0.69	0.00	0.00	0.00	0.00
	3	0.25	0.44	0.60	0.71	0.25	0.44	0.60	0.71	0.00	0.00	0.00	0.00
$ER_{150,0.03}$	1	0.13	0.24	0.34	0.43	0.13	0.24	0.34	0.43	0.00	0.00	0.00	0.00
	2	0.22	0.39	0.53	0.64	0.22	0.39	0.53	0.64	0.00	0.00	0.00	0.00
	3	0.28	0.48	0.64	0.75	0.28	0.48	0.63	0.74	0.00	0.00	0.02	0.01
	4	0.29	0.51	0.67	0.77	0.29	0.51	0.66	0.77	0.00	0.00	0.01	0.00
	5	0.29	0.51	0.67	0.78	0.29	0.51	0.66	0.77	0.00	0.00	0.01	0.01
$ER_{150,0.10}$	1	0.12	0.22	0.30	0.38	0.12	0.22	0.30	0.38	0.00	0.00	0.00	0.00
	2	0.16	0.30	0.40	0.49	0.16	0.30	0.40	0.49	0.00	0.00	0.00	0.00
	3	0.16	0.30	0.41	0.50	0.16	0.30	0.41	0.50	0.00	0.00	0.00	0.00
$ER_{200,0.03}$	1	0.10	0.19	0.27	0.34	0.10	0.19	0.27	0.34	0.00	0.00	0.00	0.00
	2	0.18	0.32	0.44	0.55	0.18	0.32	0.44	0.54	0.00	0.00	0.00	0.02
	3	0.22	0.39	0.52	0.63	0.22	0.38	0.51	0.63	0.00	0.03	0.02	0.00
	4	0.22	0.40	0.53	0.64	0.22	0.39	0.53	0.64	0.00	0.03	0.00	0.00
	5	0.22	0.40	0.53	0.64	0.22	0.39	0.53	0.64	0.00	0.03	0.00	0.00
$ER_{200,0.10}$	1	0.08	0.14	0.21	0.26	0.08	0.14	0.21	0.26	0.00	0.00	0.00	0.00
	2	0.10	0.19	0.27	0.35	0.10	0.19	0.27	0.35	0.00	0.00	0.00	0.00
	3	0.10	0.19	0.28	0.35	0.10	0.19	0.27	0.35	0.00	0.00	0.04	0.00

7.3.3 Large-Scale Networks

Up to this point, we have tested performance of Algorithm 2 on small size networks and observed that it is really timesaving even for small n values. Now we investigate running time of the algorithm on some real networks when the size of the network is large.

Firstly, we use Facebook network [74] consisting 4,039 vertices and 88,234 edges. We have already reported preprocessing times for this network. Total computational times (in s) to find a group with high $NGB^k(M)$ value is reported in Table 7.7 for each g value. We can see that as g increases, computational time increases as well.

Table 7.7: The preprocessing and total computational time (in s) to find a group with high $NGB^k(M)$ value of size g on the Facebook network

	Prep. Time	**Total Time (in s) to Find a Group With High $NGB^k(M)$ Value of Size g**									
	(in s) to	**Group Size g**									
k	**Compute PB^k**	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
1	3278.94	25.19	52.48	77.08	151.57	139.96	509.08	227.34	641.24	667.95	282.30
2	3359.18	82.07	139.66	178.79	243.31	323.52	381.36	461.24	538.17	480.67	571.51
3	3657.87	78.30	140.66	195.37	262.72	321.75	380.53	299.68	187.10	175.23	190.29
4	4227.36	35.20	50.38	75.29	118.82	204.30	380.20	182.70	498.83	379.20	515.39
5	4569.89	80.53	144.56	197.27	253.57	313.66	373.91	209.35	454.62	554.10	611.18
6	4634.01	78.94	139.60	197.91	259.76	317.56	381.07	434.81	476.74	400.30	511.55
7	4670.12	79.89	141.11	204.83	265.34	326.24	387.38	432.82	499.94	565.50	630.52
8	4679.23	81.29	137.73	198.94	84.71	90.76	106.12	121.26	137.43	153.37	168.61

* Prep.: Preprocessing

Second real-life network we consider is an online social network called the soc-anybeat [71] network. This network consists of 12.6K vertices and 67.1K edges.

For this network, the σ and d matrices are computed in about 27 hours. The matrix PB^k is computed once for each k value, and the time to compute PB^k for each k value is given in Table 7.8 in the column titled "Prep. time (in s) to Compute PB^k". Total computational times (in s) to find a group with high $NGB^k(M)$ value is reported in the following columns. We obtain similar results with the Facebook

network. Also, we can see that once the preprocessing is done, Algorithm 2 has reasonable running times even for such a large network.

Table 7.8: The preprocessing and total computational time (in s) to find a group with high $NGB^k(M)$ value of size g on the soc-anybeat network

		Total Time (in s) to Find a Group With High $NGB^k(M)$ Value of Size g									
	Prep. Time (in s) to	**Group Size g**									
k	**Compute PB^k**	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
1	56269.83	242.03	497.74	691.84	911.69	1164.70	1312.60	1706.46	1893.90	2170.47	2279.56
2	60082.31	241.53	482.21	686.31	970.54	1081.01	1348.12	1650.74	1812.17	1949.80	2300.35
3	64827.38	252.79	493.10	728.53	905.77	1138.05	1433.14	1652.48	1907.86	2104.35	2308.12
4	62537.79	264.75	500.34	755.16	951.68	1130.63	1481.37	1612.02	1945.06	2076.06	2378.45
5	61806.81	263.70	500.41	750.76	954.99	1217.70	1343.38	1675.71	1804.39	1944.78	2273.33
6	61657.36	269.01	521.08	755.85	933.26	1170.02	1459.74	1697.52	1898.57	2112.04	2151.10
7	61601.78	270.15	503.59	706.03	955.51	1243.33	1416.61	1615.55	1757.91	2131.18	2390.81
8	61567.22	267.18	505.87	737.30	1005.15	1237.29	1492.66	1726.75	1893.30	2117.30	2354.60
9	61577.45	265.88	503.18	712.87	988.18	1256.49	1412.90	1632.15	1958.30	2126.18	2405.51
10	61593.02	268.10	525.07	775.41	1006.51	1256.03	1382.37	1692.53	1959.77	2201.08	2364.59

* Prep.: Preprocessing

CHAPTER 8

CONCLUSION AND FUTURE STUDY DIRECTIONS

In this thesis, we propose variants of the classical betweenness and group betweenness centralities; namely, the k-step BC and k-step GBC, respectively. The normalized k-step BC of a vertex measures the likelihood that the vertex will get the information communicated between a randomly chosen pair of vertices through a randomly chosen shortest path within the first k steps of the start of the communication. The normalized k-step GBC of a group of vertices, similarly, measures the likelihood that information will be obtained by at least one member of the group within the first k steps. The newly introduced centrality measures may find uses in applications where it is important or critical to obtain the information within a fixed time of the start of the communication.

For the newly introduced centrality measures, we propose an algorithm to compute successively the k-step GBC of several groups of vertices. This algorithm is a modification of the one proposed by Puzis et al. [8] for the computation of the GBC of groups of vertices. We have shown that the proposed algorithm can evaluate the k-step GBC of several groups of vertices within reasonable times even for large scale networks. We also propose a greedy heuristic algorithm to find a group of vertices of size g whose k-step GBC value is high. We show that heuristic algorithm is really time-saving and has small optimality gaps. Therefore, it could be a good alternative for large scale networks or when group size g is too big to enumerate all possible subsets for a given graph. Other than these we propose a mixed integer programming formulation to compute the optimum k-step GBC value of size g which is a modification of the model proposed by Veremyev et al. in [9]. We observe that MIP formulation can compute the optimum k-step GBC value in reasonable times, espe-

cially for sparse graphs. Also, it allows to work on bigger group sizes. Note that Algorithm 1 and the MIP model may find uses in different contexts, and therefore are in a sense complementary algorithms for the k-step GBC computations.

We have made the following observations through our computational experiments.

- For a fixed g, $NGB_g^{k*}(G)$ values increase with decreasing marginal changes until some k value and remain constant thereafter.
- For a fixed k, $NGB_g^{k*}(G)$ values increase as g increases as expected.
- The k-step normalized GBC values are non-increasing as the density increases.
- Given a group size g, the group maximizing the GBC may be different from the one maximizing the k-step GBC. Moreover, the group maximizing the k-step GBC value may change from k to k. Therefore, if the user aims to obtain the information within k steps with a higher probability, it is better to use the k-step GBC measure instead of the classical GBC measure.
- When we investigate the normalized k-step GBC values of all subsets of vertices of a given size, we have seen that the distribution of the values is right-skewed. Therefore, in order to estimate the maximum k-step GBC value, one may need to sample several groups of vertices. The Algorithm 1 proposed in this study may serve this purpose, as once the preprocessing step is done, it can evaluate the k-step GBC of several groups of vertices very fast.
- For a fixed g, as k increases, the time to compute all $NGB^k(C)$ values increases as well. Moreover, for a fixed k, as g increases, the time to compute all $NGB^k(C)$ values also increases.

For further research, Algorithm 1 can be integrated with a sampling algorithm to estimate the maximum k-step GBC value for large scale networks. Another important direction for further research is to find a group of vertices minimizing the expected time the information is obtained. No study has been conducted on this problem. Moreover, in this study we assume that communication happens equally likely between every pair of vertices. An algorithm to compute k-step GBC when communication is not equally likely between every pair of vertices can be constructed.

REFERENCES

[1] K. Das, S. Samanta, and M. Pal, "Study on centrality measures in social networks: A survey," *Social Network Analysis and Mining*, vol. 8, no. 1, p. 13, 2018.

[2] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, vol. 40, no. 1, pp. 35–41, 1977.

[3] L. C. Freeman, "Centrality in social networks conceptual clarification," *Social Networks*, vol. 1, no. 3, pp. 215–239, 1978-1979.

[4] U. Brandes, "A faster algorithm for betweenness centrality," *The Journal of Mathematical Sociology*, vol. 25, no. 2, pp. 163–177, 2001.

[5] E. D. Kolaczyk, D. B. Chua, and M. Barthélemy, "Group betweenness and co-betweenness: Inter-related notions of coalition centrality," *Social Networks*, vol. 31, no. 3, pp. 190–203, 2009.

[6] M. G. Everett and S. P. Borgatti, "The centrality of groups and classes," *The Journal of Mathematical Sociology*, vol. 23, no. 3, pp. 181–201, 1999.

[7] R. Puzis, Y. Elovici, and S. Dolev, "Finding the most prominent group in complex networks," *AI Communications*, vol. 20, no. 4, pp. 287–296, 2007.

[8] R. Puzis, Y. Elovici, and S. Dolev, "Fast algorithm for successive computation of group betweenness centrality," *Physical Review E*, vol. 76, no. 5, p. 056709, 2007.

[9] A. Veremyev, O. A. Prokopyev, and E. L. Pasiliao, "Finding groups with maximum betweenness centrality," *Optimization Methods and Software*, vol. 32, no. 2, pp. 369–399, 2017.

[10] S. Dolev, Y. Elovici, R. Puzis, and P. Zilberman, "Incremental deployment of network monitors based on group betweenness centrality," *Information Processing Letters*, vol. 109, no. 20, pp. 1172–1176, 2009.

[11] W. Chen, W. Lu, and N. Zhang, "Time-critical influence maximization in social networks with time-delayed diffusion process," in *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, pp. 592–598, 2012.

[12] B. Liu, G. Cong, D. Xu, and Y. Zeng, "Time constrained influence maximization in social networks," in *2012 IEEE 12th International Conference on Data Mining*, pp. 439–448, 2012.

[13] E. Otte and R. Rousseau, "Social network analysis: a powerful strategy, also for the information sciences," *Journal of information Science*, vol. 28, no. 6, pp. 441–453, 2002.

[14] C. Haythornthwaite, "Social network analysis: An approach and technique for the study of information exchange," *Library & Information Science Research*, vol. 18, no. 4, pp. 323–342, 1996.

[15] L. Freeman, "The development of social network analysis," *A Study in the Sociology of Science*, vol. 1, 2004.

[16] D. Z. Levin and R. Cross, "The strength of weak ties you can trust: The mediating role of trust in effective knowledge transfer," *Management Science*, vol. 50, no. 11, pp. 1477–1490, 2004.

[17] D. Meltzer, J. Chung, P. Khalili, E. Marlow, V. Arora, G. Schumock, and R. Burt, "Exploring the use of social network methods in designing healthcare quality improvement teams," *Social Science & Medicine*, vol. 71, no. 6, pp. 1119–1130, 2010.

[18] W. Li, X. Wang, and M. Yu, "A research on collaboration knowledge construction in the virtual learning community by social network analysis," in *2010 International Conference on Educational and Information Technology*, vol. 2, pp. V2–323, IEEE, 2010.

[19] S. Korkmaz and A. Singh, "Impact of team characteristics in learning sustainable built environment practices," *Journal of professional issues in engineering education and practice*, vol. 138, no. 4, pp. 289–295, 2011.

[20] M. K. Di Marco, J. E. Taylor, and P. Alin, "Emergence and role of cultural boundary spanners in global engineering project networks," *Journal of Management in Engineering*, vol. 26, no. 3, pp. 123–132, 2010.

[21] X. Ruan, E. Ochieng, A. Price, and C. Egbu, "Knowledge integration process in construction projects : A social network analysis approach to compare competitive and collaborative working," *Construction Management and Economics*, vol. 30, no. 1, pp. 5–19, 2012.

[22] L. Zhang, J. He, and S. Zhou, "Sharing tacit knowledge for integrated project team flexibility: Case study of integrated project delivery," *Journal of Construction Engineering and Management*, vol. 139, no. 7, pp. 795–804, 2013.

[23] P. Chinowsky, J. Diekmann, and V. Galotti, "Social network model of construction," *Journal of Construction Engineering and Management*, vol. 134, no. 10, pp. 804–812, 2008.

[24] M. Akhavan Farshchi and M. Brown, "Social networks and knowledge creation in the built environment: A case study," *Structural Survey*, vol. 29, no. 3, pp. 221–243, 2011.

[25] A. Bavelas, "A mathematical model for group structures," *Human Organization*, vol. 7, no. 3, pp. 16–30, 1948.

[26] H. J. Leavitt, "Some effects of certain communication patterns on group performance," *Journal of Abnormal Psychology*, vol. 46, no. 1, pp. 38–50, 1951.

[27] M. M. Cohn BS, "Networks and centres of integration in Indian civilization," *Journal of Social Research*, vol. 1, no. 1, pp. 1–9, 1958.

[28] F. R. Pitts, "A graph theoretic approach to historical geography," *The Professional Geographer*, vol. 17, no. 5, pp. 15–20, 1965.

[29] M. A. Beauchamp, "An improved index of centrality," *Behavioral Science*, vol. 10, no. 2, pp. 161–163, 1965.

[30] J. A. Czepiel, "Word-of-mouth processes in the diffusion of a major technological innovation," *Journal of Marketing Research*, vol. 11, no. 2, pp. 172–180, 1974.

[31] D. L. Rogers, "Sociometric analysis of interorganizational relations: Application of theory and measurement," *Rural Sociology*, vol. 39, no. 4, pp. 487–503, 1974.

[32] N. Coles, "Analysing serious crime groups as social networks," *The British Journal of Criminology*, vol. 41, no. 4, pp. 580–594, 2001.

[33] J. Bruun and E. Brewe, "Talking and learning physics: Predicting future grades from network measures and force concept inventory pretest scores," *Physical Review Physics Education Research*, vol. 9, no. 2, p. 020109, 2013.

[34] M. E. Shaw, "Group structure and the behavior of individuals in small groups," *The Journal of Psychology*, vol. 38, no. 1, pp. 139–149, 1954.

[35] J. Nieminen, "On the centrality in a graph," *Scandinavian Journal of Psychology*, vol. 15, no. 1, pp. 332–336, 1974.

[36] G. Sabidussi, "The centrality index of a graph," *Psychometrika*, vol. 31, no. 4, pp. 581–603, 1966.

[37] M. E. J. Newman, "Scientific collaboration networks. I. Network construction and fundamental results," *Physical Review E*, vol. 64, no. 1, p. 016131, 2001.

[38] P. Bonacich, "Factoring and weighting approaches to status scores and clique identification," *The Journal of Mathematical Sociology*, vol. 2, no. 1, pp. 113–120, 1972.

[39] P. Bonacich, "Some unique properties of eigenvector centrality," *Social Networks*, vol. 29, no. 4, pp. 555–564, 2007.

[40] G. Yan, T. Zhou, B. Hu, Z.-Q. Fu, and B. Wang, "Efficient routing on complex networks," *Physical Review. E*, vol. 73, no. 4, p. 046108, 2006.

[41] N. Jayaweera, K. Perera, and J. Munasinghe, "Centrality measures to identify traffic congestion on road networks: A case study of Sri Lanka," *IOSR Joournal of Mathematics*, vol. 13, no. 2, pp. 13–19, 2017.

[42] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review E*, vol. 69, no. 2, p. 026113, 2004.

[43] X. Liu, J. Bollen, M. L. Nelson, and H. V. de Sompel, "Co-authorship networks in the digital library research community," *Information Processing & Management*, vol. 41, no. 6, pp. 1462–1480, 2005.

[44] J.-G. Liu, Z.-G. Xuan, Y.-Z. Dang, Q. Guo, and Z.-T. Wang, "Weighted network properties of Chinese nature science basic research," *Physica A: Statistical Mechanics and its Applications*, vol. 377, no. 1, pp. 302–314, 2007.

[45] M. K. Sparrow, "The application of network analysis to criminal intelligence: An assessment of the prospects," *Social Networks*, vol. 13, no. 3, pp. 251–274, 1991.

[46] D. Z. Grunspan, B. L. Wiggins, and S. M. Goodreau, "Understanding classrooms through social network analysis: A primer for social network analysis in education research," *CBE—Life Sciences Education*, vol. 13, no. 2, pp. 167–178, 2014.

[47] N. Magaia, A. P. Francisco, P. Pereira, and M. Correia, "Betweenness centrality in delay tolerant networks: A survey," *Ad Hoc Networks*, vol. 33, pp. 284–305, 2015.

[48] U. Brandes, "On variants of shortest-path betweenness centrality and their generic computation," *Social Networks*, vol. 30, no. 2, pp. 136–145, 2008.

[49] S. Borgatti and M. Everett, "A graph-theoretic perspective on centrality," *Social Networks*, vol. 28, no. 4, pp. 466–484, 2006.

[50] A. Shimbel, "Structural parameters of communication networks," *The Bulletin of Mathematical Biophysics*, vol. 15, no. 4, pp. 501–507, 1953.

[51] P. L. Szczepański, T. P. Michalak, and T. Rahwan, "Efficient algorithms for game-theoretic betweenness centrality," *Artificial Intelligence*, vol. 231, pp. 39 – 63, 2016.

[52] M. J. Newman, "A measure of betweenness centrality based on random walks," *Social Networks*, vol. 27, pp. 39–54, 2003.

[53] T. Opsahl, F. Agneessens, and J. Skvoretz, "Node centrality in weighted networks: Generalizing degree and shortest paths," *Social Networks*, vol. 32, no. 3, pp. 245 – 251, 2010.

[54] R. Geisberger, P. Sanders, and D. Schultes, "Better approximation of betweenness centrality," in *Proceedings of the Meeting on Algorithm Engineering & Expermiments*, pp. 90–100, Society for Industrial and Applied Mathematics, 2008.

[55] M. Lujak and S. Giordani, "Centrality measures for evacuation: Finding agile evacuation routes," *Future Generation Computer Systems*, vol. 83, pp. 401 – 412, 2018.

[56] R. Grassi, F. Calderoni, M. Bianchi, and A. Torriero, "Betweenness to assess leaders in criminal networks: New evidence using the dual projection approach," *Social Networks*, vol. 56, pp. 23 – 32, 2019.

[57] T. Hayashi, T. Akiba, and Y. Yoshida, "Fully dynamic betweenness centrality maintenance on massive networks," *The Proceedings of then Very Large Data Bases Endowment*, vol. 9, no. 2, pp. 48–59, 2015.

[58] M.-J. Lee, S. Choi, and C.-W. Chung, "Efficient algorithms for updating betweenness centrality in fully dynamic graphs," *Information Sciences*, vol. 326, pp. 278 – 296, 2016.

[59] M. Nasre, M. Pontecorvi, and V. Ramachandran, "Betweenness centrality–incremental and faster," in *International Symposium on Mathematical Foundations of Computer Science*, pp. 577–588, Springer, 2014.

[60] U. Brandes and C. Pich, "Centrality estimation in large networks," *International Journal of Bifurcation and Chaos*, vol. 17, no. 07, pp. 2303–2318, 2007.

[61] M. Riondato and E. M. Kornaropoulos, "Fast approximation of betweenness centrality through sampling," *Data Mining and Knowledge Discovery*, vol. 30, no. 2, pp. 438–475, 2016.

[62] D. A. Bader, S. Kintali, K. Madduri, and M. Mihail, "Approximating betweenness centrality," in *Algorithms and Models for the Web-Graph* (A. Bonato and F. R. K. Chung, eds.), pp. 124–137, Springer Berlin Heidelberg, 2007.

[63] M. Hinne, "Local approximation of centrality measures," *Computing*, 2011.

[64] M. Rysz, F. M. Pajouh, and E. L. Pasiliao, "Finding clique clusters with the highest betweenness centrality," *European Journal of Operational Research*, vol. 271, no. 1, pp. 155 – 164, 2018.

[65] C. Ni, C. R. Sugimoto, and J. Jian, "Degree, Closeness, and Betweenness: Application of group centrality measurements to explore macro-disciplinary evolution diachronically," in *Proceedings of the 13th International Society of Scientometrics and Informetrics Conference (ISSI)*, pp. 1–13, 2011.

[66] A. Kchiche and F. Kamoun, "Access-points deployment for vehicular networks based on group centrality," *2009 3rd International Conference on New Technologies, Mobility and Security*, pp. 1–6, 2009.

[67] R. Puzis, Y. Altshuler, Y. Elovici, S. Bekhor, Y. Shiftan, and A. S. Pentland, "Augmented betweenness centrality for environmentally aware traffic monitoring in transportation networks," *Journal of Intelligent Transportation Systems*, vol. 17, no. 1, pp. 91–105, 2013.

[68] M. Tubi, R. Puzis, and Y. Elovici, "Deployment of DNIDS in social networks," in *2007 IEEE Intelligence and Security Informatics*, pp. 59–65, 2007.

[69] R. Puzis, M. Tubi, Y. Elovici, C. Glezer, and S. Dolev, "A decision support system for placement of intrusion detection and prevention devices in large-scale networks," *ACM Transactions on Modeling and Computer Simulation*, vol. 22, no. 1, pp. 5:1–5:26, 2011.

[70] J. Guan, Z. Yan, S. Yao, C. Xu, and H. Zhang, "GBC-based caching function group selection algorithm for SINET," *Journal of Network and Computer Applications*, vol. 85, no. 1, pp. 56 – 63, 2017.

[71] R. A. Rossi and N. K. Ahmed, "The network data repository with interactive graph analytics and visualization." http://networkrepository.com, 2015. Accessed 27 February 2019.

[72] R. Willoughby, "The university of Florida sparse matrix collection." https://sparse.tamu.edu, 1970. Accessed 27 February 2019.

[73] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[74] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection." http://snap.stanford.edu/data, 2014. Accessed 27 February 2019.

[75] D. S. Hochbaum and A. Pathria, "Analysis of the greedy approach in problems of maximum k-coverage," *Naval Research Logistics*, vol. 45, no. 6, pp. 615–627, 1998.

APPENDIX A

APPENDIX

A.1 Single-source Shortest-paths Problem

Algorithm 3 SSSP

1: Input: $G = (V, E)$, $s \in V$
2: Output: σ_{sv} and $d(s, v),\ v \in V$
3: $Q \leftarrow$ empty queue.
4: $S \leftarrow$ empty stack.
5: $Pred[w] \leftarrow \{\}, w \in V$
6: $\sigma_{st} \leftarrow 0$, $d_{st} \leftarrow \infty, t \in V$
7: $\sigma_{ss} \leftarrow 1, d_{ss} \leftarrow 0$
8: Add vertex s to the queue, $s \rightarrow Q$
9: **while** Q is not empty **do**
10: Take off v from the queue, $v \leftarrow Q$.
11: Add v to the stack, $v \rightarrow S$.
12: **for all** vertex w such that $\{v, w\} \in E$ **do**
13: **if** $d(s, w) = \infty$ **then**
14: $d(s, w) \leftarrow d(s, v) + 1$
15: Add vertex w to the queue, $w \rightarrow Q$
16: **else if** $d(s, w) = d(s, v) + 1$ **then**
17: $\sigma_{sw} \leftarrow \sigma_{sw} + \sigma_{sv}$
18: Append v to the predecessor's of w , $v \rightarrow Pred[w]$
19: **end if**
20: **end for**
21: **end while**

A.2 Path List Creating Algorithm

Algorithm 4 Path List Creating Algorithm

Inputs: $G = (V, E)$

Output: $Path(r, s, t, k)$ for any $s, t \in V$, $r = 1, \ldots, \sigma_{st}$, $k = 1, \ldots, d(s, t) - 1$

for $v \in V$ **do**
  $L \leftarrow \max_{t \in V} d(v, t)$
  for $\ell = 2$ to L **do**
    for $s \in V_\ell(v)$ **do**
      $pathindex_{vs} \leftarrow 1$
      for $t \in V_1(s) \cap V_{\ell-1}(v)$ **do**
        for $pathindex_{vt} = 1$ to σ_{vt} **do**
          for $nodeindex = 1$ to $\ell - 2$ **do**
            $Path(pathindex_{vs}, v, s, nodeindex) \leftarrow$
            $Path(pathindex_{vt}, v, t, nodeindex)$
          end for
          $Path(pathindex_{vs}, v, s, \ell - 1) \leftarrow t$
          $pathindex_{vs} \leftarrow pathindex_{vs} + 1$
        end for
      end for
    end for
  end for
end for

A.3 Absolute and Relative Differences on Erdös Renyi Graphs

Table A.1: The maximum absolute differences between $NGB_g^{k*}(G)$ and $NGB^k(C_g^*)$ for different k and g values on 50 randomly generated Erdös-Renyi graphs of order 20

Graph	**g** \ **k**	**1**	**2**	**3**	**4**	**5**
$ER_{20,0.15}$	1	5.94	5.02	2.06	0.00	0.00
	2	7.54	12.37	9.08	1.58	0.00
	3	7.46	5.69	2.50	0.00	0.26
	4	8.13	3.77	4.63	1.07	0.00
	5	9.48	6.32	0.48	0.00	0.00
	6	11.84	3.38	0.09	0.00	0.00
$ER_{20,0.20}$	1	3.39	1.96	0.20		
	2	6.84	0.42	1.14		
	3	7.46	1.62	0.36		
	4	9.13	3.87	0.29		
	5	6.52	3.80	0.35		
	6	5.94	3.42	0.05		
$ER_{20,0.25}$	1	4.47	2.40	0.00		
	2	5.74	1.11	0.00		
	3	2.16	1.13	0.00		
	4	3.84	1.87	0.00		
	5	3.84	3.21	0.50		
	6	5.50	0.93	0.18		

Table A.2: The maximum relative differences between $NGB_g^{k*}(G)$ and $NGB^k(C_g^*)$ for different k and g values on 50 randomly generated Erdös-Renyi graphs of order 20

Graph	**g** \ **k**	**1**	**2**	**3**	**4**	**5**
$ER_{20,0.15}$	1	39.85	21.83	6.58	0.00	0.00
	2	22.31	32.87	17.88	2.52	0.00
	3	14.90	8.82	3.28	0.00	0.34
	4	13.64	5.02	5.75	1.28	0.00
	5	15.60	7.35	0.53	0.00	0.00
	6	17.75	3.79	0.10	0.00	0.00
$ER_{20,0.20}$	1	20.52	6.97	0.76		
	2	20.00	0.99	1.96		
	3	16.58	2.42	0.58		
	4	16.67	4.68	0.35		
	5	9.52	4.49	0.39		
	6	7.74	3.77	0.06		
$ER_{20,0.25}$	1	24.19	8.47	0.00		
	2	20.34	2.50	0.00		
	3	4.83	2.09	0.00		
	4	5.85	2.50	0.00		
	5	5.64	3.98	0.59		
	6	7.44	1.04	0.20		

Table A.3: The maximum absolute differences between $NGB_g^{k*}(G)$ and $NGB^k(C_g^*)$ for different k and g values on 50 randomly generated Erdös-Renyi graphs of order 40

Graph	**g** \ **k**	**1**	**2**	**3**	**4**	**5**	**6**
$ER_{40,0.06}$	1	4.67	4.48	3.46	0.77	0.00	0.00
	2	6.70	2.75	2.90	1.30	2.95	0.00
	3	10.13	5.26	4.23	3.25	2.95	0.00
	4	5.56	6.19	5.33	5.15	4.10	2.18
	5	9.01	9.36	4.24	1.99	0.51	0.07
	6	7.77	8.32	5.98	0.98	0.34	0.01
$ER_{40,0.08}$	1	2.06	1.61	0.50	0.00	0.00	
	2	4.20	1.36	0.72	0.04	0.00	
	3	5.25	2.21	1.88	0.90	0.03	
	4	4.31	2.06	1.64	0.67	0.00	
	5	5.47	3.85	3.35	0.59	0.02	
	6	6.21	2.04	1.22	0.23	0.00	
$ER_{40,0.1}$	1	1.63	2.50	0.66	0.00		
	2	2.46	0.94	0.10	0.00		
	3	4.25	1.74	0.73	0.08		
	4	4.34	0.88	0.64	0.00		
	5	4.34	3.05	1.56	0.00		
	6	4.26	1.65	0.32	0.00		

Table A.4: The maximum relative differences between $NGB_g^{k*}(G)$ and $NGB^k(C_g^*)$ for different k and g values on 50 randomly generated Erdös-Renyi graphs of order 40

Graph	**g** \ **k**	**1**	**2**	**3**	**4**	**5**	**6**
$ER_{40,0.06}$	1	57.72	29.72	13.81	2.69	0.00	0.00
	2	38.62	8.62	5.09	2.02	4.17	0.00
	3	43.76	12.61	6.49	4.41	3.79	0.00
	4	17.42	11.36	7.19	6.48	5.01	2.60
	5	22.35	14.77	5.15	2.47	0.60	0.09
	6	17.79	11.69	6.94	1.12	0.41	0.01
$ER_{40,0.08}$	1	20.31	12.03	2.96	0.00	0.00	
	2	22.42	4.66	1.77	0.09	0.00	
	3	19.59	3.97	3.93	1.78	0.04	
	4	12.96	3.31	2.70	1.04	0.00	
	5	14.83	6.22	4.55	0.75	0.02	
	6	14.53	2.60	1.57	0.28	0.00	
$ER_{40,0.1}$	1	22.76	17.17	3.15	0.00		
	2	13.97	3.37	0.37	0.00		
	3	18.28	4.29	1.41	0.16		
	4	12.21	1.59	1.01	0.00		
	5	10.52	5.33	2.36	0.00		
	6	8.68	2.38	0.41	0.00		

Table A.5: The maximum absolute differences between $NGB_g^{k*}(G)$ and $NGB^k(C_g^*)$ for different k and g values on 50 randomly generated Erdös-Renyi graphs of order 60

Graph	**g** \ **k**	**1**	**2**	**3**	**4**	**5**	**6**
$ER_{60,0.06}$	1	6.73	3.37	1.68	1.47	0.00	0.00
	2	4.36	4.57	2.15	0.67	1.66	0.67
	3	6.20	7.25	4.31	5.17	1.38	0.70
	4	5.78	4.09	6.56	5.04	0.47	0.00
	5	7.42	6.31	7.20	4.90	0.89	0.07
$ER_{60,0.08}$	1	5.12	4.31	1.19	0.12	0.00	0.00
	2	5.12	1.66	0.89	0.48	0.00	0.00
	3	5.12	1.63	0.93	0.23	0.00	0.00
	4	6.76	1.91	0.58	0.15	0.00	0.00
	5	7.47	3.34	2.01	1.84	0.67	0.03
$ER_{60,0.1}$	1	1.08	1.07	0.34			
	2	1.12	0.25	0.05			
	3	1.18	1.18	0.30			
	4	1.79	0.86	0.16			
	5	2.26	1.50	0.08			

Table A.6: The maximum relative differences between $NGB_g^{k*}(G)$ and $NGB^k(C_g^*)$ for different k and g values on 50 randomly generated Erdös-Renyi graphs of order 60

Graph	**g** \ **k**	**1**	**2**	**3**	**4**	**5**	**6**
$ER_{60,0.06}$	1	82.93	54.17	8.30	0.55	0.00	0.00
	2	39.97	6.58	2.88	1.51	0.00	0.00
	3	27.72	4.98	2.28	0.51	0.00	0.00
	4	30.71	4.48	1.05	0.28	0.00	0.00
	5	28.49	6.25	3.09	2.68	0.95	0.05
$ER_{60,0.08}$	1	23.33	12.89	2.28			
	2	11.37	1.39	0.18			
	3	6.50	4.51	0.98			
	4	7.32	2.52	0.41			
	5	8.00	3.05	0.15			
$ER_{60,0.1}$	1	12.94	9.45	0.27			
	2	4.71	2.60	0.00			
	3	5.07	2.58	0.29			
	4	6.68	0.97	0.00			
	5	3.92	1.11	0.06			

A.4 Distribution of the $NGB^k(C)$ Values of All Subsets C of V on Randomly Generated Graphs

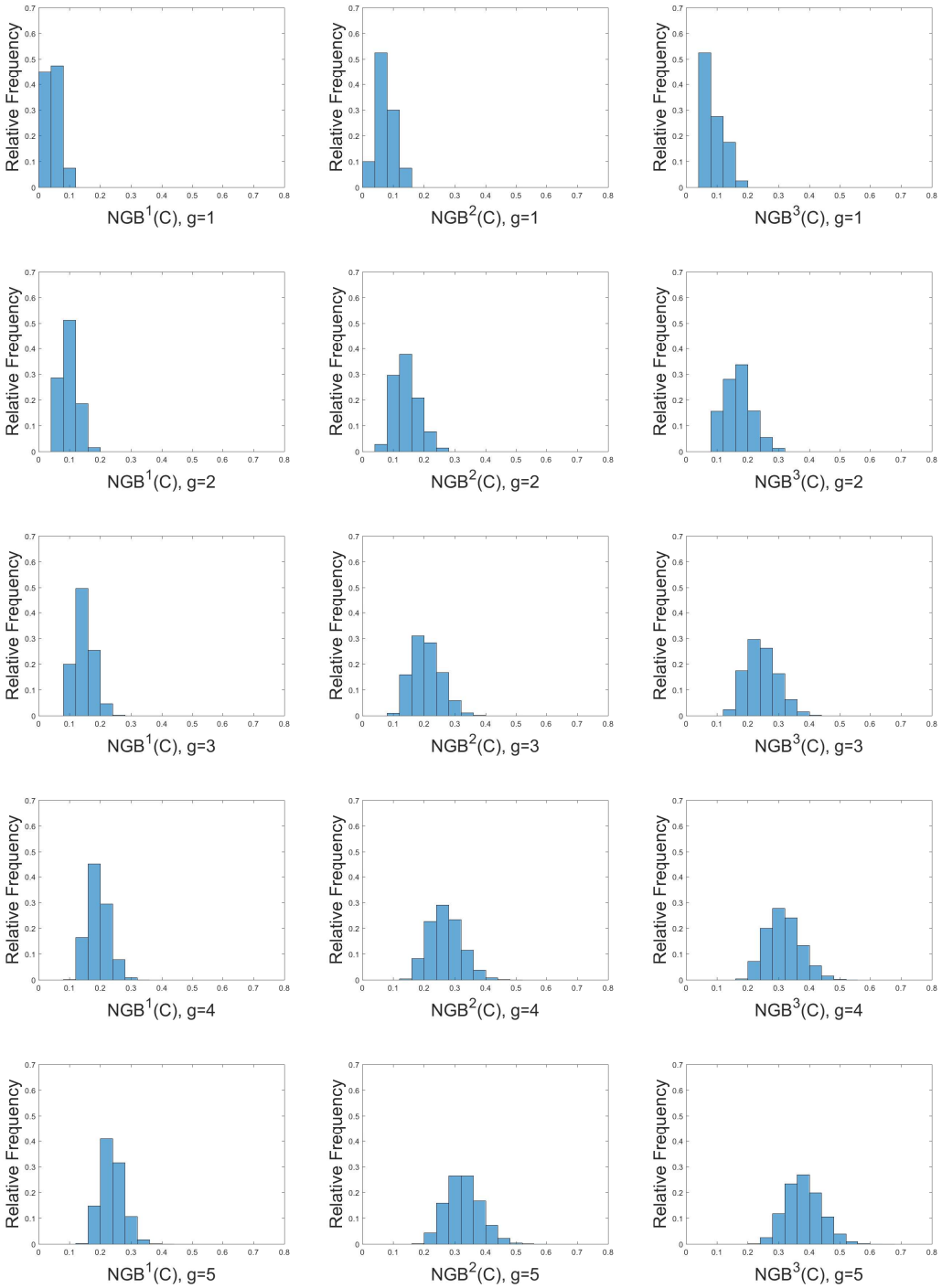


Figure A.1: Distribution of the $NGB^k(C)$ values of all subsets C of V for different k and g values on the graph $ER_{40,01}$

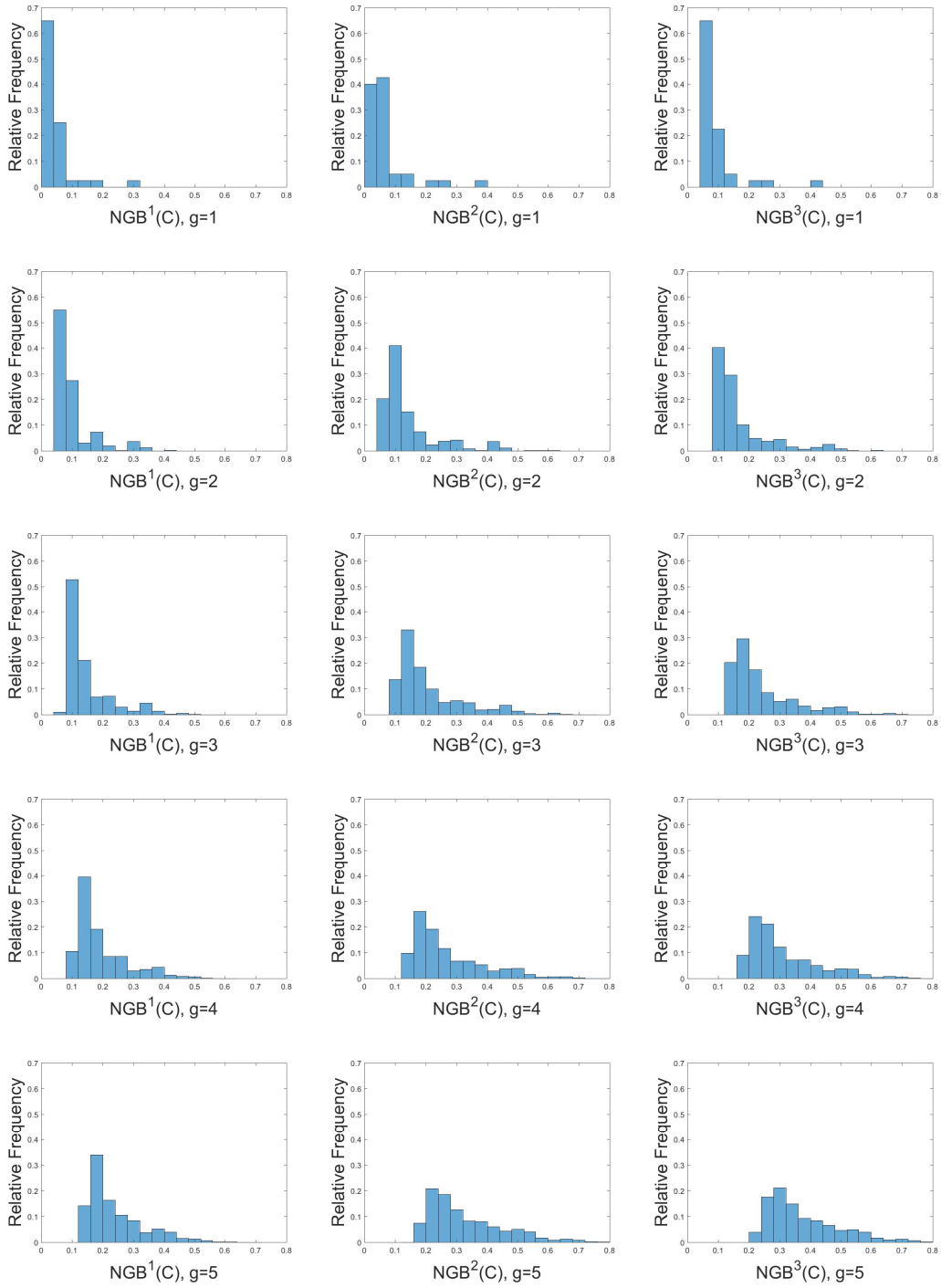


Figure A.2: Distribution of the $NGB^k(C)$ values of all subsets C of V for different k and g values on the graph $PA_{40,2}$