UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

DISTRIBUTED ESTIMATION OF CENTRALITY MEASURES IN COMPLEX NETWORKS

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

DOCTOR OF PHILOSOPHY

By

WEI WANG
Norman, Oklahoma
2016

DISTRIBUTED ESTIMATION OF CENTRALITY MEASURES
IN COMPLEX NETWORKS

A DISSERTATION APPROVED FOR THE
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

BY

Dr. Choon Yik Tang, Chair

Dr. John Jiang

Dr. S. Lakshmivarahan

Dr. Thordur Runolfsson

Dr. Krishnaiyan Thulasiraman

To my family

Acknowledgements

First and foremost, I wish to express my sincere gratitude to my advisor, Dr. Choon Yik Tang, for his guidance and unconditional support throughout all the years of my graduate studies, and for being an excellent mentor. During the past four years, he provided me a precious opportunity to learn how an idea is created and developed. He was gracious and generous to offer his knowledge and his experience to me while working on this research project, and I have acquired many skills that are crucial to my future career. I am truly grateful for having had the opportunity to work closely with him.

I am also grateful to Dr. Lipei Huang from Tsinghua University and Dr. John Jiang, for recommending me to Dr. Tang as a Ph.D. student. I would not have been able to work with Dr. Tang if it were not for their help.

I am also thankful to Dr. S. Lakshmivarahan, Dr. Thordur Runolfsson, and Dr. Krishnaiyan Thulasiraman for their time and effort in serving on my graduate committee. Their invaluable suggestions and recommendations have significantly improved the quality of my research and dissertation.

I would also like to acknowledge the National Science Foundation for their generous financial support.

Finally, I wish to thank my parents, Liulin Wang and Ping Wei. Without their selfless love, support, and encouragement, I would not have a chance of achieving all this.

Table of Contents

List of Tables

List of Figures

Abstract

Knowing how important a node or an edge is, within a network, can be very valuable, for instance, in identifying those who are susceptible to malicious attacks, those who are bottlenecks to network performances, and those who are better suited as leaders, so that preventive measures may be taken, resources may be properly allocated, and roles may be properly assigned.

In this dissertation, we develop a novel collection of scalable distributed algorithms, which enable nodes in a large-scale network to cooperatively learn how important they are individually, with only local interaction and without any global coordination nor knowledge of the network topology. The node importance, or criticality, will be measured using the most fundamental centrality measures from the area of complex networks, namely, the betweenness centrality, closeness centrality, as well as a subset of their variations—those regarded by the network science community to be most fundamental that we believe are distributedly computable using a dynamical systems approach—so that insights gained in computing them shed light on how to compute other measures of similar nature.

The algorithms are developed based on tools and ideas from dynamical systems, graph theory, and network science. For each centrality measure, we first introduce some variables with graph-theoretic meaning, and expressed each measure as a function of these variables. Afterwards, we derive a set of equality and inequality constraints on these variables that characterize each centrality measure in lieu of its original definition. Every constraint involves only variables that are associated with

neighboring nodes, so that neighboring nodes can check whether they are satisfied. Therefore, all of these constraints are distributed in nature.

Next, we use these constraints to develop a scalable distributed algorithm, which enables nodes in a network to cooperatively estimate their individual centrality with only local interaction and without any centralized coordination, nor high memory usages. Specifically, for tree graphs, the introduced variables are linearly and non-singularly related. Thus, by turning these constraints into a state equation, and the centrality measure function into an output equation, we subsequently obtain a networked dynamical system describing a distributed algorithm. For general graphs, the constraints we have discovered are necessary but insufficient, so the problem cannot be solved exactly. Therefore, we formulate a distributed optimization problem, a regularized linear program to estimate the centrality measures over the network by using a gradient method. Taking the gradient of the objective function with respect to the optimization variables, we obtain a scalable continuous-time distributed algorithm.

Moreover, for tree graphs, we show that each algorithm is a networked dynamical system, whose affine state equation has a unique equilibrium point that is always exponentially or finite-time stable, and whose output equation at the equilibrium point always yields the unknown centrality measure, thereby solving the problem. For general graphs, we evaluate the performance of the algorithm via extensive simulation, showing that it yields fairly accurate estimates in terms of ordering, on random geometric, Erdős-Rényi, and Barabási-Albert graphs.

Finally, we experiment our algorithms for estimating node and edge betweenness centralities on both computer generated graphs and real networks for community detection and information spreading. We also propose a method for better spreading with the knowledge of community structures. The method using estimated betweenness performs very well in almost all scenarios.

Chapter 1

Introduction

1.1 Background and Motivation

In many applications of networks, nodes and edges within the network have different levels of importance. Due to where they are located in the network, and who they are connected with, some nodes and edges are inherently more critical to the network's well-being than others. For example, in a power grid, failure of certain power generators and transmission lines are more likely to trigger cascading effects; in a wireless ad hoc network, certain communication devices and links are more effective to route data; in a transportation system, certain junctions and roads are more critical to traffic conditions; and in a social network, different people and websites have different levels of influence and popularity based on who they know, and which websites they are linked to.

Often, knowing how important individual nodes and edges are within the network can be very valuable, as they allow for preventive measures to be taken, resources to be properly allocated, and roles to be properly assigned. When the network is small, identifying which nodes and edges are important, and understanding why they are so, by, say, humans who oversee or study the network, is not a difficult task. How-

Figure 1.1: A portion of the Internet (adopted from [1]).

ever, when the network has an enormous number of nodes and edges with extremely complicated connections—such as many real networks that exist today, including the Internet, the U.S. power grid, and biological networks—the task becomes very challenging, even for humans with help from powerful computers. Moreover, with the continued increase in the size and complexity of networks and the rising demand for their autonomy, it is becoming increasingly desirable that nodes and edges themselves can carry out such a task, as opposed to relying entirely on humans.

In the area of complex networks [3,4], a growing set of statistical measures known as *centrality measures*, which assign to each node or edge a score representing its importance, have been introduced (see [5] for a survey). In this dissertation, we consider two of the oldest and most fundamental ones, namely, *node betweenness* and *classic closeness* described below, as well as a subset of their variations.

Node Betweenness: Given an undirected and connected network or graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, \ldots, N\}$ denotes the set of $N \geq 2$ nodes and $\mathcal{E} \subset \{\{i, j\} : i, j \in \mathcal{V}, i \neq j\}$ the set of edges, the *node betweenness* B_i of a node

$i \in \mathcal{V}$ is defined in Anthonisse [6] and Freeman [7] as

$$B_i \triangleq \sum_{\substack{k \in \mathcal{V} \\ k \neq i}} \sum_{\substack{\ell \in \mathcal{V} \\ \ell \neq i,k}} \frac{\sigma(k,\ell,i)}{\sigma(k,\ell)}, \tag{1.1}$$

where $\sigma(k,\ell)$ is the number of shortest paths from nodes k to ℓ, and $\sigma(k,\ell,i)$ is the number of those that go through node i. Thus, the larger the number of shortest paths node i lies on, the higher its B_i. It follows that B_i attempts to measure how "strategically located" node i is within the network, making it one of the most fundamental centrality measures [3]. In fact, node betweenness and its counterpart called *edge betweenness* (to be defined in Chapter 2) have found broad applications in such areas as power systems [8], transportation systems [9], bioinformatics [10], and bibliometrics [11]. Conceivably, it can also be used in network security, since nodes with high B_i's are more susceptible to malicious attacks, and in communication networks, since such nodes are more likely bottlenecks to data routing. It can further be used to improve network topology: if node i has a high B_i and its neighbors k and ℓ have low B_k and B_ℓ, then adding an edge between nodes k and ℓ would reduce B_i because all the shortest paths that go through nodes k, i, ℓ could then bypass node i.

Classic Closeness: Another fundamental centrality measure, defined in Bavelas [12] and Sabidussi [13], is the *classic closeness* (often referred to simply as *closeness*) C_i of a node $i \in \mathcal{V}$, defined as

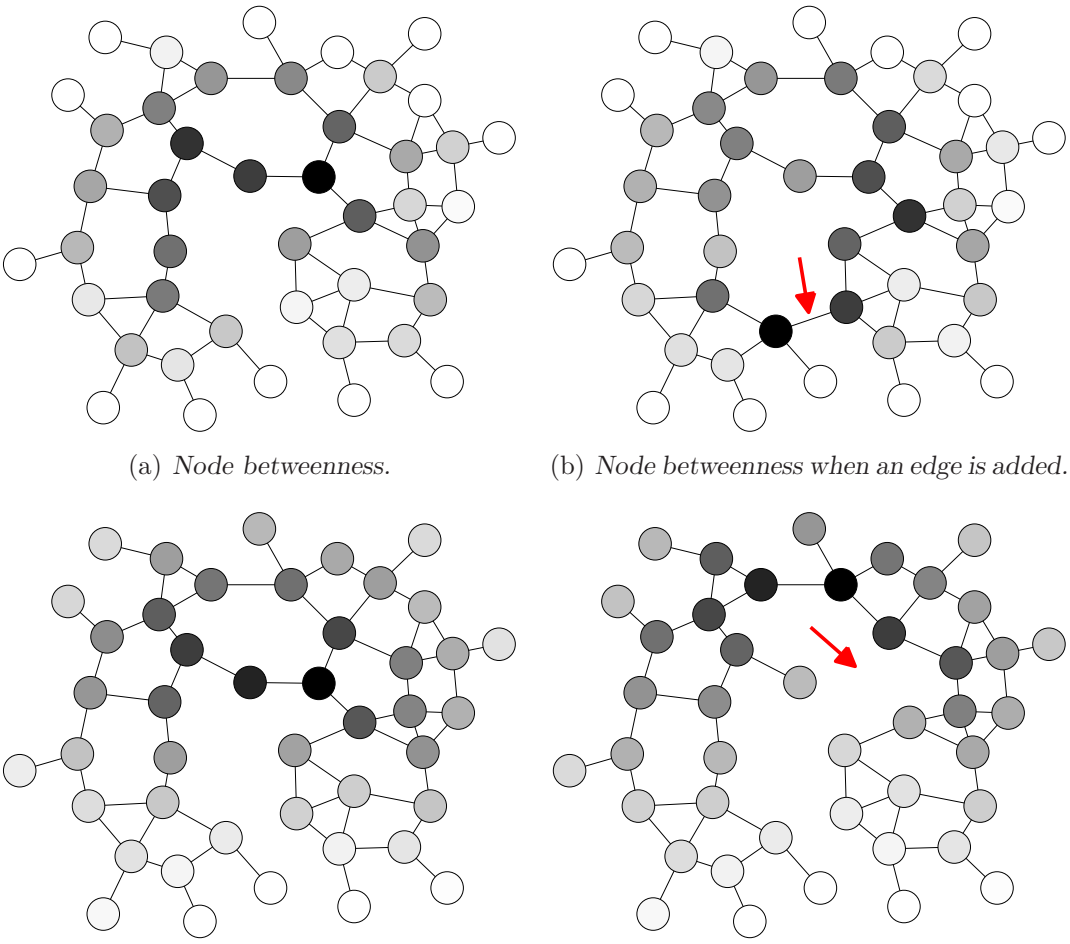
$$C_i \triangleq \frac{N-1}{\sum_{j \in \mathcal{V}} d_{ij}}, \tag{1.2}$$

where d_{ij} is the distance (i.e., length of the shortest paths) from nodes i to j, and the factor $N-1$ is inserted so that $C_i \in (0,1]$. Hence, the closer node i is,

on average, to all other nodes, the higher its C_i. It follows that nodes with high C_i's are effective in, for instance, spreading diseases or rumors to, and gathering information from, everyone else. Indeed, classic closeness has been applied to numerous areas, including epidemiology [14], social networks [15], and power systems [16, 17]. It can also be used to, say, find travel directions: if graph $\mathcal{G}$ represents a city and its nodes represent different parts of the city, then a traveler who has access to local closeness but not the city map might be able to get to the city center by following a path along which the closeness keeps increasing (much like a gradient method in optimization). Therefore, both node betweenness and classic closeness are intriguing centrality measures that have significant untapped potential. An illustration of these two measures, which highlights their key attributes, is provided in Figure 1.2.

1.2 Literature Review

Although node betweenness and classic closeness are useful indicators of node importance in a complex network, their computation may be prohibitively difficult because both of them require the construction of shortest paths between *all* pairs of nodes. While there are several well-established algorithms for constructing shortest paths (e.g., the Floyd-Warshall algorithm [18] and the Johnson's algorithm [19]), these algorithms were developed nearly half a century ago and thus were not designed to handle networks of such a scale. To overcome this limitation, a few algorithms aimed at computing node betweenness in large networks have been proposed in the literature (e.g., Freeman [20], Brandes [21], Kolaczyk [22]). These algorithms, however, are *centralized* in nature, requiring that all the information about the graph $\mathcal{G}$ be available at one place, at one time, in order to execute. Such a requirement, unfortunately, is often difficult to meet in a large network for various reasons, including security and



(a) *Node betweenness.* (b) *Node betweenness when an edge is added.*

(c) *Classic closeness.* (d) *Classic closeness when a node is deleted.*

Figure 1.2: The importance of nodes in a network can be quite different even among neighbors (see (a)–(d), in which the darker a node, the more important it is). The importance of nodes can also change significantly when a different centrality measure is used (compare (a) with (c)), and when an edge or a node is added or deleted elsewhere (compare (a) with (b), and (c) with (d)).

privacy concerns and storage and single-point failure issues.

The drawback facing these centralized algorithms raises a question: is it possible to develop fully distributed algorithms that enable nodes in a complex network to cooperatively learn about their own importance as measured by node betweenness, classic closeness, and other notable centrality measures? The availability of such algorithms would produce fundamental advances in the area of complex networks, allowing us to better engineer large-scale networked systems and better understand large-scale networked systems that occur in nature. For the former, several engineering examples have been given in Section 1.1. For the latter, an example is to run such distributed algorithms on parallel computers to more rapidly analyze a huge biological network with millions of nodes and edges. Another example is a social network that can be decomposed into multiple subgraphs, each of which is owned by a company. The companies wish to determine the importance of nodes they own, but are unwilling to share their subgraphs with others due to security and privacy concerns. If the companies agree to run such distributed algorithms, they would be able to achieve their goals without having to publicize their subgraphs.

Although the above question is of significant interest, it has not been addressed other than in a 2003 technical report by Lehmann and Kaufmann [23], which introduced distributed algorithms for computing betweenness and closeness centralities. The algorithms in [23], however, are not really "distributed" in the sense of the controls community. Specifically, each algorithm in [23] is memory intensive, demanding that every node stores a list of messages that grows with the network size. Moreover, each of them is non-homogeneous, requiring that nodes act differently depending on what they have received, and which phase the algorithm is in. More seriously, each of the algorithms benefits only *one* node per execution: only the leader node that initiates the execution can determine its centrality score; all other nodes cannot determine theirs and are there just to "help." In contrast, distributed algorithms from

the controls community are memory non-intensive, homogeneous (i.e., same update rules for every node at every time), and benefit all nodes in one execution (e.g., in distributed averaging, all nodes can gradually compute the average).

At present, the controls community has developed a large (and growing) collection of distributed algorithms for performing *distributed consensus/computation/optimization*. In distributed consensus, where the primary objective is for nodes in a network to achieve a consensus or agreement, many variations of the basic problem have been studied, including problems in continuous-time [24–35] and in discrete-time with synchronous [24, 26, 29–33, 36–53] and asynchronous [42, 54–72] time models. In addition, the basic problem has also been extended to represent a variety of applications, including motion coordination [73], vehicle formation [74, 75], and flocking [65, 76, 77], to name a few. In distributed computation, a number of specific problems have been studied to date, including distributed averaging which is often addressed in the context of distributed consensus, and problems where the goal is to compute the solution to a system of linear equations [78–82], the maximum [56, 59, 83–85], the sum/count [36, 55, 56, 84], and the power mean [56, 83, 86]. Other types of problems have also been addressed, such as the computation of distributed redistribution [37], design of distributed Kalman filters [34,78,87–89], as well as computation of linear functions [50,90–92], average-max-min [25], log-sum-exp [85], and a class of general functions [83, 86, 93], and many more. Finally, in distributed optimization, research efforts have led to, for instance, the family of incremental subgradient algorithms [94–102], non-incremental ones [53, 103–110].

Given the successful development of a rich collection of distributed consensus/computation/optimization algorithms—most of which are based on a dynamical systems approach—and given the essentially unexplored area of distributed computation of betweenness and closeness centralities, an intriguing question is: is it possible to use the same dynamical systems approach to distributedly compute those centralities?

This dissertation is devoted to addressing this question.

1.3 Original Contributions and Dissertation Outline

In this dissertation, we develop a novel collection of simple and scalable distributed algorithms, which enable nodes in a large-scale network to cooperatively learn how important or critical they are individually, with only local interaction and without any global coordination nor knowledge of the network topology. The node importance, or criticality, will be measured using the most fundamental centrality measures from the area of complex networks, namely, the betweenness centrality, closeness centrality, as well as a subset of their variations, such as edge betweenness and exponential closeness. An outline of the dissertation, along with its original contributions, is provided below.

In Chapter 2, we consider the distributed computation of *node* and *edge betweenness* that characterize how often a node or edge lies on the shortest paths between all pairs of nodes. For each measure, we construct dynamical systems approaches to develop several synchronous (continuous- and discrete-time) and asynchronous distributed algorithms, which enable every node in an undirected and unweighted tree graph to compute its own measure with only local interaction and without any centralized coordination. We show that the algorithms are simple and scalable, with the synchronous continuous-time algorithm being unconditionally exponentially convergent, the synchronous discrete-time algorithm unconditionally exhibiting a deadbeat response, and the asynchronous algorithm being asymptotically stable. Moreover, we show that these algorithms require minimal node memories to execute, bypass entirely the need to construct shortest paths. The algorithms are applicable only to tree graphs.

In Chapter 3, we address the distributed computation of *classic closeness* and *exponential closeness*, which differ in how the distances are taken into account. For each variant, we construct continuous- and discrete-time distributed algorithms, with which nodes in an undirected and unweighted tree graph can cooperatively determine their own closeness by talking only to neighbors, executing simple homogeneous update rules, and consuming minimal physical memories. We show that each algorithm is a networked dynamical system, whose affine state equation has a unique equilibrium point that is always exponentially or finite-time stable, and whose output equation at the equilibrium point always yields the unknown closeness, thereby solving the problem. The algorithms are applicable only to tree graphs.

In Chapter 4, we develop a scalable distributed algorithm, which enables every node in a network to estimate its own betweenness and the betweenness of edges incident on it with only local interaction and without any centralized coordination, nor high memory usages. The development is based on exploiting various local properties of shortest paths, and on formulating and solving an unconstrained distributed optimization problem. We also evaluate the performance of the algorithm via simulation on a number of random geometric graphs, showing that it yields betweenness estimates that are fairly accurate in terms of ordering.

In Chapter 5, we develop a scalable distributed algorithm, which enables every node in a network to estimate its own closeness. We first derive a set of linear inequality and equality constraints, which are distributed in nature, that characterize closeness centrality in lieu of its original definition. We then use these constraints to develop a scalable distributed algorithm, which enables nodes in a network to cooperatively estimate their individual closeness with only local interaction and without any centralized coordination, nor high memory usages. Finally, we evaluate the performance of the algorithm via extensive simulation, showing that it yields closeness estimates that are 91% accurate in terms of ordering, on random geometric, Erdős-

Rényi, and Barabási-Albert graphs.

In Chapter 6, we test our algorithms for estimating node and edge betweenness centralities on both computer generated graphs and real networks for community detection and information spreading, and then propose a method for a better spreading information with the knowledge of community structures. These real network data sets are maintained by Dr. Tim Davis of Texas A&M University and Dr. Yifan Hu of Yahoo Labs. The data sets are available at http://www.cise.ufl.edu/research/sparse/matrices [111]. Based on the evaluation results, we notice that with the knowledge of community structures, information could be spread faster in the network by correctly detecting influential people, and the method using estimated betweenness performs very well in almost all scenarios.

Finally, in Chapter 7, we conclude the dissertation with several remarks and provide a number of possible future research directions.

Chapter 2

Distributed Computation of Betweenness on Tree Graphs

2.1 Introduction

In many applications of networks, nodes and edges within a network have different importance. Due to where they are located in the network, and who they are connected with, some nodes and edges are inherently more critical to the network's well-being than others. Thus, knowing how important a node or an edge is, by itself or by its neighbors, can be very valuable.

In the area of complex networks, a growing set of statistical measures referred to as *centrality measures*, which assign to each node or edge a number representing its importance, have been proposed. In this chapter, we consider the distributed computation of two such measures, namely, *node* and *edge betweenness*. To facilitate the development, we define each of these measures below in the context of an undirected, unweighted, and connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, \ldots, N\}$ denotes the set of $N \geq 2$ nodes and $\mathcal{E} \subset \{\{i, j\} : i, j \in \mathcal{V}, i \neq j\}$ denotes the set of edges:

- The *node betweenness* B_i of a node $i \in \mathcal{V}$ is first conceived in Anthonisse [6]

and later defined in Freeman [7] in (1.1) as

$$B_i \triangleq \sum_{\substack{k\in\mathcal{V}\\ k\neq i}} \sum_{\substack{\ell\in\mathcal{V}\\ \ell\neq i,k}} \frac{\sigma(k,\ell,i)}{\sigma(k,\ell)},$$

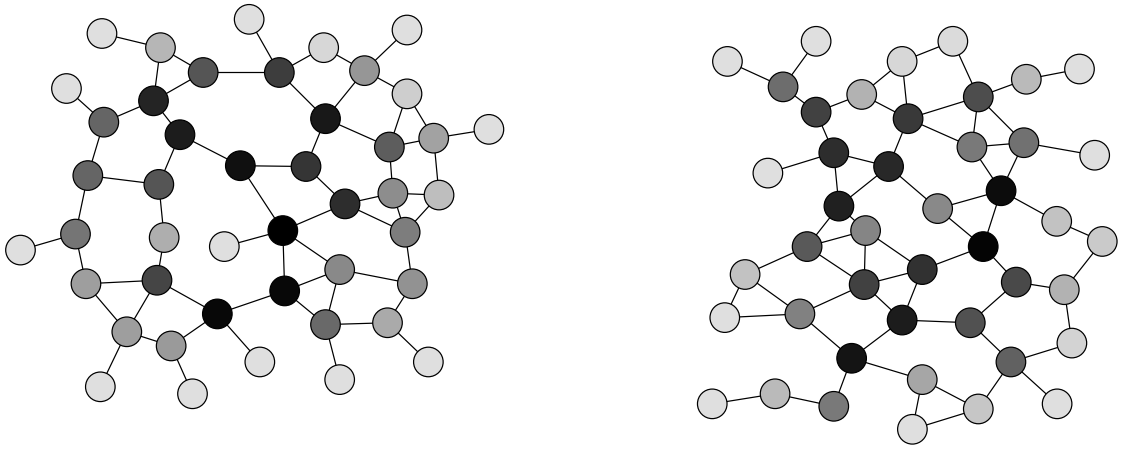
where $\sigma(k,\ell)$ is the number of shortest paths from nodes k to ℓ, and $\sigma(k,\ell,i)$ is the number of those that go through node i. Hence, the larger the number of shortest paths node i lies on, the higher its B_i. It follows that B_i attempts to measure how "strategically located" node i is within graph $\mathcal{G}$. Figure 2.2(a)–(c) illustrates the notion of node betweenness, in which the darker a node i, the higher its B_i. The B_i's of nodes in a network can be quite different even among neighbors, and can change significantly when network structure changed somewhere, such as a bridge is added or deleted.

- Analogous to B_i, the *edge betweenness* $B_{\{i,j\}}$ of an edge $\{i,j\} \in \mathcal{E}$ is defined in [7] is obtained by replacing $\sigma(k,\ell,i)$ in the definition of B_i in (1.1) by $\sigma(k,\ell,\{i,j\})$, the *edge betweenness* $B_{\{i,j\}}$ of an edge $\{i,j\} \in \mathcal{E}$ as

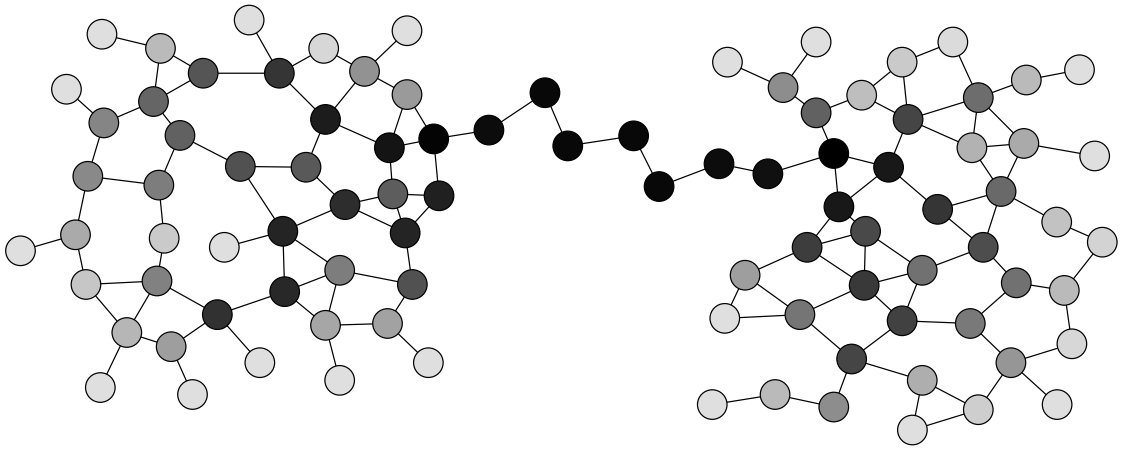
$$B_{\{i,j\}} \triangleq \sum_{k\in\mathcal{V}} \sum_{\substack{\ell\in\mathcal{V}\\ \ell\neq k}} \frac{\sigma(k,\ell,\{i,j\})}{\sigma(k,\ell)}, \tag{2.1}$$

  where $\sigma(k,\ell,\{i,j\})$ is the number of shortest paths from nodes k to ℓ that go through edge $\{i,j\}$. Therefore, $B_{\{i,j\}}$ is a measure of how strategically located edge $\{i,j\}$ is within graph $\mathcal{G}$, which may represent, say, a road in a city, or a transmission line in a power grid. Figure 2.2(a)–(c) depicts the notion of edge betweenness, showing that it parallels that of node betweenness, and also can change significantly when network structure changed somewhere.

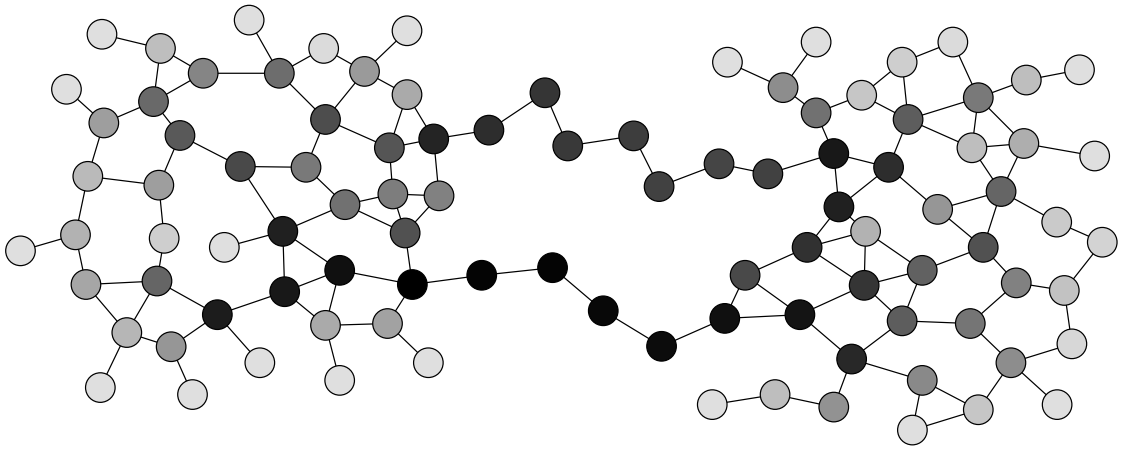
Although node and edge betweenness are useful centrality measures, their computation may be difficult because they are defined in terms of all the shortest paths.



(a) Original graphs.

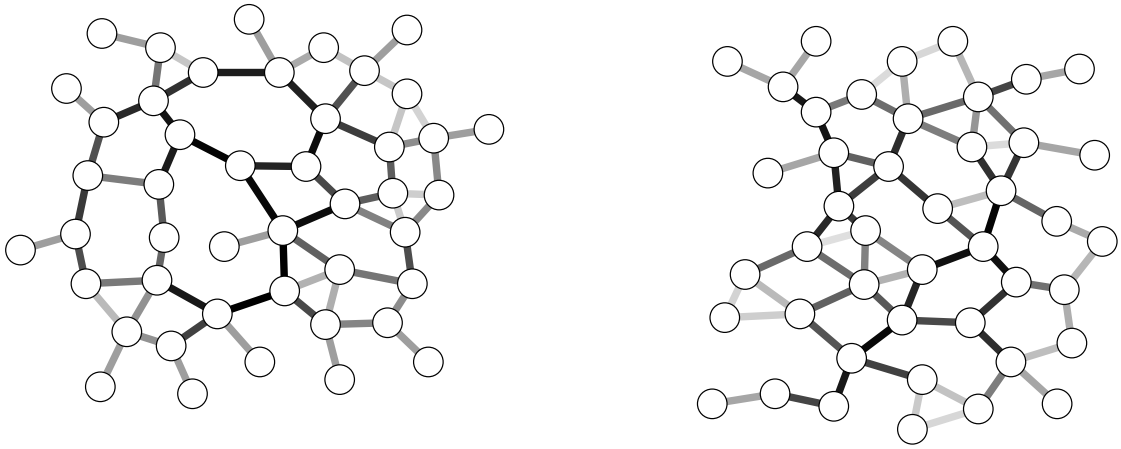


(b) Adding a bridge.

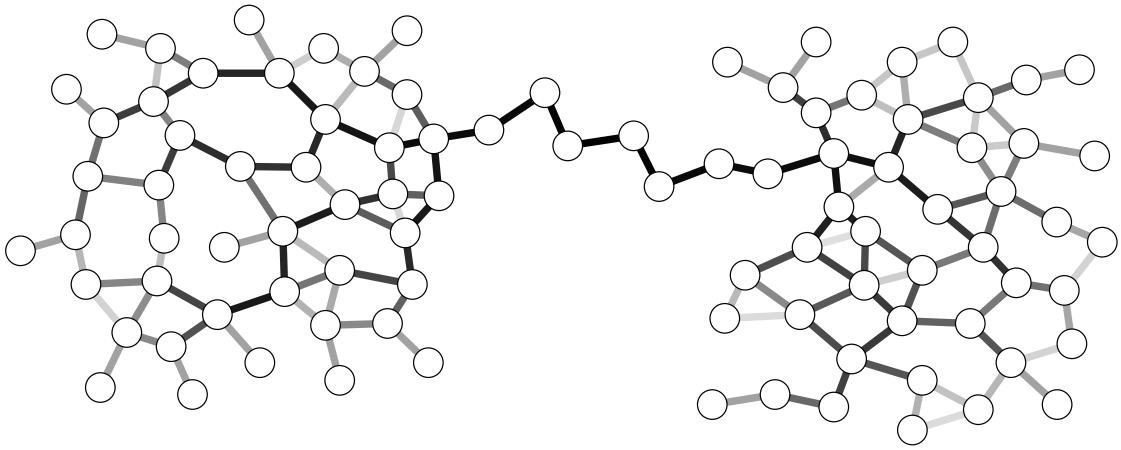


(c) Adding another bridge.

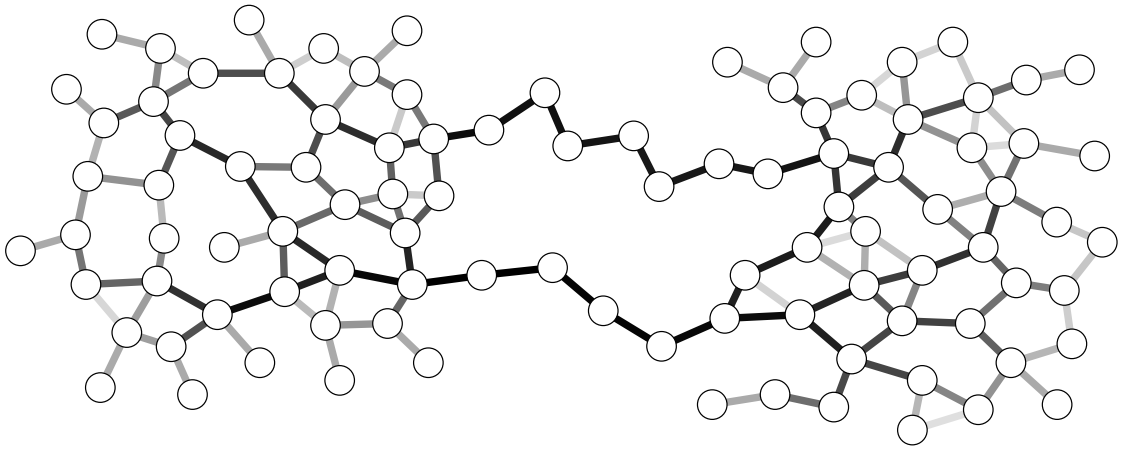
Figure 2.1: An illustration of node betweenness.



(a) Original graphs.



(b) Adding a bridge.



(c) Adding another bridge.

Figure 2.2: An illustration of edge betweenness.

While there are several well-established algorithms [18, 19] for constructing shortest paths, and a few algorithms [20–22] aimed at computing node betweenness in large networks, these algorithms, however, are centralized in nature, requiring that all the information about the graph $\mathcal{G}$ be available at one place, at one time, in order to execute. Such a requirement, unfortunately, is often difficult to meet in a large network for various reasons, including security and privacy concerns as well as storage and single-point failure issues.

Motivated by the aforementioned consideration and by successful development of a rich collection of distributed consensus/computation/optimization algorithms (e.g., [24,30,51,53,54,57,72,110], to name just a few), in this chapter we consider the distributed computation of node and edge betweenness. We show that if the graph $\mathcal{G}$ is a *tree*, it is possible to construct synchronous (continuous- and discrete-time) and asynchronous distributed algorithms, which enable each node $i \in \mathcal{V}$ in the tree $\mathcal{G}$ to compute its own node betweenness B_i and its incident edge betweenness $B_{\{i,j\}}$ for every $j \in \mathcal{N}_i = \{j \in \mathcal{V} : \{i, j\} \in \mathcal{E}\}$, with only local interaction and without any centralized coordination. In systems-theoretic terms, the algorithms are networked dynamical systems with affine state equations and nonlinear output equations, in which each node maintains a small subset of the state and output variables and updates them homogeneously. We show that each of the algorithms has a unique equilibrium point in the state space, at which the corresponding outputs are the unknown B_i's and $B_{\{i,j\}}$'s. Moreover, for synchronous algorithms, the unique equilibrium point for the continuous-time algorithm is unconditionally exponentially stable, with a convergence rate that is independent of the tree $\mathcal{G}$ (as measured by the eigenvalues of the system matrix). In contrast, the unique equilibrium point for the discrete-time algorithm is unconditionally finite-time stable, with the finite convergence time coinciding with the diameter of $\mathcal{G}$, denoted as $D(\mathcal{G})$ (i.e., the system exhibits a deadbeat response), and can be further reduced by half (i.e., $\lceil \frac{D(\mathcal{G})}{2} \rceil$) when the network size N

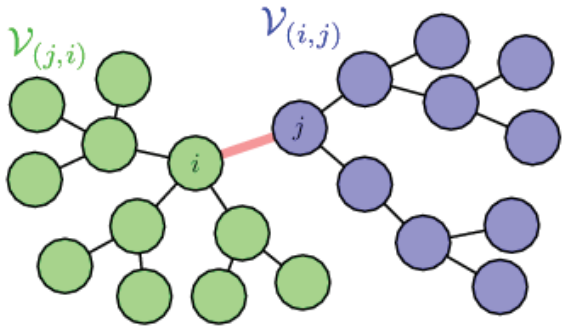


Figure 2.3: A graphical illustration of the sets $\mathcal{V}_{(i,j)}$ and $\mathcal{V}_{(j,i)}$.

is known. Furthermore, the unique equilibrium point for asynchronous algorithm is asymptotically stable. Finally, as can be seen from their development, the algorithms are simple and scalable, require minimal node memories to execute, bypass entirely the need to construct shortest paths, and can handle time-varying topologies (as long as $\mathcal{G}$ remains a tree).

The outline of this chapter is as follows: Section 2.2 introduces some preliminaries. Section 2.3 derives a number of algebraic relationships that are key to subsequent development. Based on them, Section 2.4 develops several synchronous (continuous- and discrete-time) and asynchronous distributed algorithms for computing node and edge betweenness. Section 2.5 presents simulation results that demonstrate the effectiveness of the synchronous (continuous- and discrete-time) algorithms. Finally, Section 2.6 concludes the chapter.

2.2 Preliminaries

Reconsider the undirected, unweighted, and connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in Section 2.1 and suppose $\mathcal{G}$ is a *tree*, i.e., it has N nodes in $\mathcal{V}$ and $N-1$ edges in $\mathcal{E}$ with no cycles. Although $\mathcal{G}$ is undirected, for the purpose of development let us associate with each edge $\{i, j\} \in \mathcal{E}$ a fictitious pair of directed edges denoted as (i, j) and (j, i) (i.e., braces are for undirected edges, while parentheses are for directed ones). In addition,

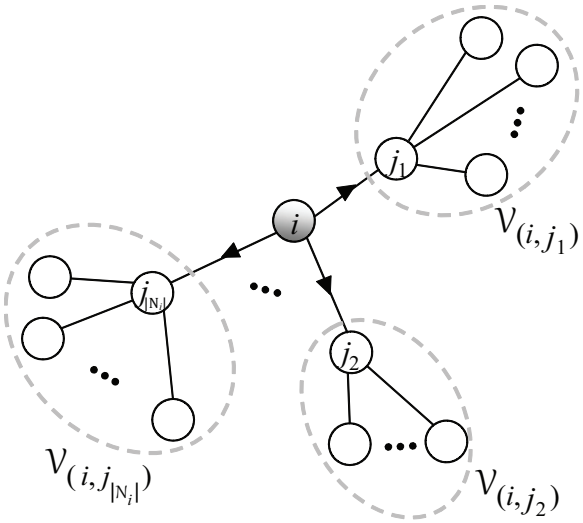


Figure 2.4: A graphical illustration of the sets $\mathcal{V}_{(i,j)}$ $\forall i \in \mathcal{V}$ $\forall j \in \mathcal{N}_i$.

let $\tilde{\mathcal{E}} = \{(i,j) : i,j \in \mathcal{V}, \{i,j\} \in \mathcal{E}\}$ denote the set of $2(N-1)$ directed edges. Since $\mathcal{G}$ is a tree, deleting any edge $\{i,j\} \in \mathcal{E}$ results in two connected components, one containing node j and the other containing node i. For convenience, let $\mathcal{V}_{(i,j)} \subset \mathcal{V}$ and $\mathcal{V}_{(j,i)} \subset \mathcal{V}$ denote, respectively, the nonempty sets of nodes in these two connected components, i.e.,

$$\begin{aligned}\mathcal{V}_{(i,j)} = \{k \in \mathcal{V} : k \text{ is in the connected component} \\ \text{containing } j \text{ if } \{i,j\} \in \mathcal{E} \text{ is deleted}\}.\end{aligned} \tag{2.2}$$

With (2.2), for each edge $\{i,j\} \in \mathcal{E}$, $\mathcal{V}$ can be partitioned into $\mathcal{V}_{(i,j)}$ and $\mathcal{V}_{(j,i)}$, i.e.,

$$\mathcal{V}_{(i,j)} \cup \mathcal{V}_{(j,i)} = \mathcal{V} \quad \text{and} \quad \mathcal{V}_{(i,j)} \cap \mathcal{V}_{(j,i)} = \emptyset. \tag{2.3}$$

Similarly, for each node $i \in \mathcal{V}$,

$$\mathcal{V} \text{ can be partitioned into } \{i\} \text{ and } \mathcal{V}_{(i,j)} \ \forall j \in \mathcal{N}_i, \tag{2.4}$$

where $\mathcal{N}_i = \{j \in \mathcal{V} : \{i,j\} \in \mathcal{E}\}$ denotes the set of neighbors of node i. Moreover, for each directed edge $(i,j) \in \tilde{\mathcal{E}}$, let $x_{(i,j)}$ denote the number of nodes in $\mathcal{V}_{(i,j)}$, i.e.,

$$x_{(i,j)} \triangleq |\mathcal{V}_{(i,j)}|, \tag{2.5}$$

where $|\cdot|$ denotes the cardinality of a set.

2.3 Algebraic Relationships

Observe that since $\mathcal{G}$ is a tree, for each pair of distinct nodes $k, \ell \in \mathcal{V}$, there is exactly one shortest path joining them, i.e.,

$$\sigma(k,\ell) = 1.$$

Thus, the node betweenness B_i of each node $i \in \mathcal{V}$ and the edge betweenness $B_{\{i,j\}}$ of each edge $\{i,j\} \in \mathcal{E}$, introduced in (1.1) and (2.1) simplify to

$$B_i = \sum_{\substack{k \in \mathcal{V} \\ k \neq i}} \sum_{\substack{\ell \in \mathcal{V} \\ \ell \neq i,k}} \sigma(k,\ell,i)$$

and

$$B_{\{i,j\}} = \sum_{k \in \mathcal{V}} \sum_{\ell \in \mathcal{V}, \ell \neq k} \sigma(k,\ell,\{i,j\}).$$

Partitioning $\mathcal{V}$ into i and $\mathcal{V}_{(i,j)}$ $\forall j \in \mathcal{N}_i$, based on (2.4), we obtain

$$B_i = \sum_{k' \in \mathcal{N}_i} \sum_{k \in \mathcal{V}_{(i,k')}} \sum_{\ell' \in \mathcal{N}_i} \sum_{\substack{\ell \in \mathcal{V}_{(i,\ell')} \\ \ell \neq k}} \sigma(k,\ell,i). \tag{2.6}$$

Partitioning $\mathcal{V}$ into $\mathcal{V}_{(i,j)}$ and $\mathcal{V}_{(j,i)}$, according to (2.3), we get

$$B_{\{i,j\}} = \sum_{k \in \mathcal{V}_{(i,j)} \cup \mathcal{V}_{(j,i)}} \sum_{\substack{\ell \in \mathcal{V}_{(i,j)} \cup \mathcal{V}_{(j,i)} \\ \ell \neq k}} \sigma(k, \ell, \{i,j\}). \tag{2.7}$$

Again, due to $\mathcal{G}$ being a tree, $\forall k' \in \mathcal{N}_i$ and $\forall \ell' \in \mathcal{N}_i$, if $k' \neq \ell'$, then $\forall k \in \mathcal{V}_{(i,k')}$ and $\forall \ell \in \mathcal{V}_{(i,\ell')}$, we have $\sigma(k, \ell, i) = 1$. Otherwise, i.e., if $k' = \ell'$, then $\forall k \in \mathcal{V}_{(i,k')}$ and $\forall \ell \in \mathcal{V}_{(i,\ell')}$ with $k \neq \ell$, we have $\sigma(k, \ell, i) = 0$. Furthermore, if $k \in \mathcal{V}_{(i,j)}$ and $\ell \in \mathcal{V}_{(j,i)}$, or if $\ell \in \mathcal{V}_{(i,j)}$ and $k \in \mathcal{V}_{(j,i)}$, then we have $\sigma(k, \ell, \{i,j\}) = 1$, since the shortest path joining nodes k and ℓ must go through edge $\{i,j\}$. Otherwise, i.e., if $k, \ell \in \mathcal{V}_{(i,j)}$ or $k, \ell \in \mathcal{V}_{(j,i)}$, then we have $\sigma(k, \ell, \{i,j\}) = 0$, since the shortest path joining nodes k and ℓ does not go through edge $\{i,j\}$. These expressions of $\sigma(k, \ell, i)$ and $\sigma(k, \ell, \{i,j\})$, along with (2.6) and (2.7), imply that

$$B_i = \sum_{j \in \mathcal{N}_i} \sum_{\substack{k \in \mathcal{N}_i \\ k \neq j}} x_{(i,j)} x_{(i,k)} \tag{2.8}$$

and

$$B_{\{i,j\}} = 2 x_{(i,j)} x_{(j,i)}. \tag{2.9}$$

With (2.8) and (2.9), we have shown that both B_i and $B_{\{i,j\}}$ originally defined in terms of the shortest paths between all pairs of nodes, may be expressed as nonlinear functions of variables $x_{(i,j)}$'s, which have simple graph-theoretic interpretation.

As it follows from (2.8) and (2.9), if each node $i \in \mathcal{V}$ knows $x_{(i,j)}$ $\forall j \in \mathcal{N}_i$, it could calculate B_i by itself. If, in addition, node i knows $x_{(j,i)}$ $\forall j \in \mathcal{N}_i$, it could also calculate $B_{\{i,j\}}$ $\forall j \in \mathcal{N}_i$. Therefore, it is desirable to have a method that enables the nodes to learn about their "local" $x_{(i,j)}$'s. To develop such a method, observe from

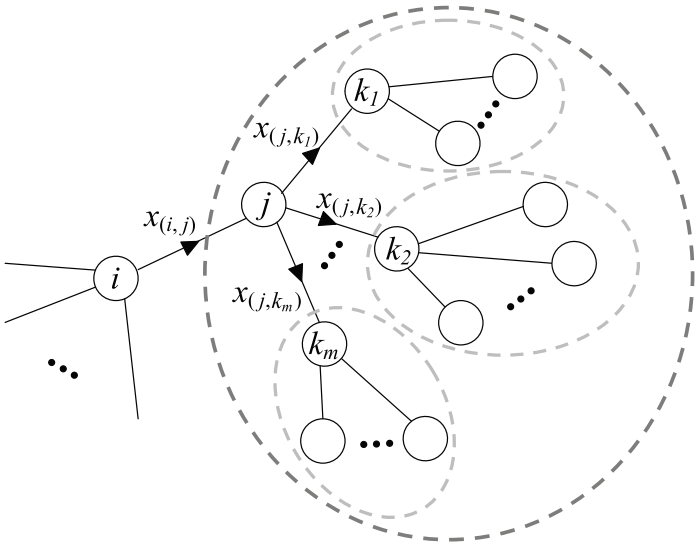


Figure 2.5: A graphical illustration of expression (2.12).

(2.3) that

$$x_{(i,j)} + x_{(j,i)} = N, \quad \forall i \in \mathcal{V}, \forall j \in \mathcal{N}_i, \tag{2.10}$$

and from (2.4) that

$$\sum_{k \in \mathcal{N}_j} x_{(j,k)} = N - 1, \quad \forall j \in \mathcal{V}. \tag{2.11}$$

Combining (2.10) and (2.11), we obtain:

$$x_{(i,j)} - \sum_{\substack{k \in \mathcal{N}_j \\ k \neq i}} x_{(j,k)} = 1, \quad \forall i \in \mathcal{V}, \forall j \in \mathcal{N}_i, \tag{2.12}$$

Expression (2.12) has a couple of implications. First, as is illustrated in Figure 2.5, expression (2.12) implies that if edge $\{i, j\} \in \mathcal{E}$ is removed from the tree $\mathcal{G}$, the number of nodes in the connected component containing node j (i.e., $x_{(i,j)}$) is equal to one (i.e., due to node j) plus the remaining number of nodes in that connected component, which happens to be equal to $\sum_{k \in \mathcal{N}_j, k \neq i} x_{(j,k)}$. Second, expression (2.12) provides

$2(N-1)$ *linear* equations relating the $2(N-1)$ unknowns $x_{(i,j)}$ $\forall (i,j) \in \tilde{\mathcal{E}}$, i.e., there are as many equations as there are unknowns. Consequently, by introducing a vector $\mathbf{x} \in \mathbb{R}^{2(N-1)}$ obtained by stacking these $2(N-1)$ unknowns $x_{(i,j)}$'s, (2.12) may be written in matrix form as

$$H\mathbf{x} = \mathbb{1}, \tag{2.13}$$

where $\mathbf{x} \in \mathbb{R}^{2(N-1)}$ is a vector containing the $x_{(i,j)}$'s, $\mathbb{1} \in \mathbb{R}^{2(N-1)}$ is the all-one column vector, and $H \in \mathbb{R}^{2(N-1)\times 2(N-1)}$ is a matrix having the following appealing properties:

Lemma 1. *The matrix H is a unipotent matrix with all its $2(N-1)$ eigenvalues at 1.*

Proof. Observe that H in (2.13) depends on the order in which the $2(N-1)$ variables $x_{(i,j)}$'s are stacked into the vector $\mathbf{x}$. Also note that such an order does not affect the eigenvalues of H because permutation of rows and columns of H may be regarded as a similarity transformation that leaves all the eigenvalues of H intact. Hence, consider without loss of generality the following rule for stacking the $2(N-1)$ variables $x_{(i,j)}$'s:

- First, arbitrarily pick a node $r \in \mathcal{V}$ and view it as a root node.
- Second, let L_0 denote the set containing only node r. Also, let $L_1 \subset \mathcal{V}$ denote the set of nodes that are *one* hop away from node r, $L_2 \subset \mathcal{V}$ the set of nodes that are *two* hops away, and so on, until the set L_p is reached, where $p = \max_{j\in\mathcal{V}} d_{rj}$ and d_{rj} is the distance between nodes r and j. Starting with an empty vector $\mathbf{x}$, insert into $\mathbf{x}$ the set of $x_{(i,j)}$ where $i \in L_0$ and $j \in L_1$, where the order among them can be arbitrary. Upon completion, insert into $\mathbf{x}$ the set of $x_{(i,j)}$ where $i \in L_1$ and $j \in L_2$. Repeat this process until $x_{(i,j)}$ for all $i \in L_{p-1}$ and $j \in L_p$ are inserted. At this point, the first $N-1$ elements of the vector $\mathbf{x}$ are defined, making up half of its length of $2(N-1)$.

- Third, for each $\ell \in \{1, 2, \ldots, N-1\}$, if the ℓth element of $\mathbf{x}$ is $x_{(i,j)}$, then let the $(\ell + N - 1)$th element of $\mathbf{x}$ be $x_{(j,i)}$.

In this fashion, the order in which the $2(N-1)$ variables $x_{(i,j)}$'s are stacked into the vector $\mathbf{x}$ is completely defined. With this ordering, it is straightforward to see that H has the form

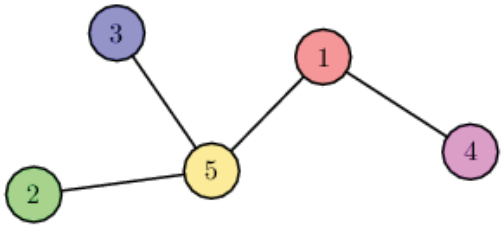
$$H = \left[\begin{array}{cccc|cccc} 1 & * & \ldots & * & 0 & 0 & \ldots & 0 \\ 0 & 1 & \ddots & \vdots & 0 & 0 & \ldots & 0 \\ \vdots & \ddots & \ddots & * & \vdots & \vdots & \ddots & \vdots \\ 0 & \ldots & 0 & 1 & 0 & 0 & \ldots & 0 \\ \hline 0 & * & \ldots & * & 1 & 0 & \ldots & 0 \\ * & 0 & \ddots & \vdots & * & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & * & \vdots & \ddots & \ddots & 0 \\ * & \ldots & * & 0 & * & \ldots & * & 1 \end{array}\right]. \tag{2.14}$$

Figure 2.6 shows an example of an network with its original node indices, unstructured H matrix, and its structured H matrix after ordering the node indices in a breadth-first manner.

Notice from (2.14) that H has a 2-by-2 block triangular structure, in which the first block on the diagonal of H is an upper triangular matrix with 1 on its diagonal, while the second block is a lower triangular matrix also with 1 on its diagonal. Thus, all the $2(N-1)$ eigenvalues of H are at 1, making it unipotent and completing the proof. □

Corollary 1. *The matrix $I - H$, where $I \in \mathbb{R}^{2(N-1)\times 2(N-1)}$ is the identity matrix, is a nilpotent matrix with all its $2(N-1)$ eigenvalues at 0.*

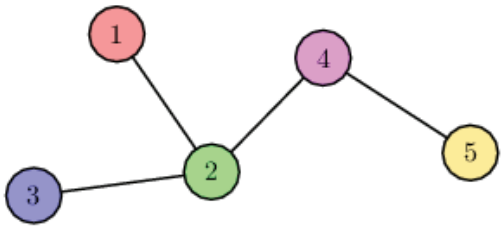
Proof. The proof is an immediate consequence of (2.14). □



(a) Original node indices.

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & -1 & -1 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 & -1 & -1 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

(b) Unstructured H matrix.



(c) Node indices after ordering.

$$H = \left[\begin{array}{cccc|cccc} 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & -1 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{array}\right]$$

(d) Structured H matrix.

Figure 2.6: An example of structuring H matrix on the 5-node tree graph.

2.4 Distributed Algorithms

In this section, we leverage the results from Section 2.3 to develop continuous-time and discrete-time synchronous distributed algorithms in Section 2.4.1–2.4.3 and an asynchronous distributed algorithm in Section 2.4.4 for computing node and edge betweenness.

2.4.1 Continuous-Time Algorithm

Recall from Section 2.3 that if each node $i \in \mathcal{V}$ is able to determine the values of $x_{(i,j)}$ and $x_{(j,i)}$ $\forall j \in \mathcal{N}_i$, then it could use (2.8) and (2.9) to compute B_i and $B_{\{i,j\}}$ $\forall j \in \mathcal{N}_i$ by itself. Also, recall from (2.13) that the $2(N-1)$ variables $x_{(i,j)}$ $\forall (i,j) \in \tilde{\mathcal{E}}$ are linearly related through H, which by Lemma 1 is unipotent with all its eigenvalues at 1. This result implies that the $2(N-1)$ linear equations (2.12) relating the $x_{(i,j)}$'s are independent and, thus, have a unique solution. More importantly, the result implies that $-H$ is always asymptotically stable with all its eigenvalues at -1 regardless of

the topology of the tree $\mathcal{G}$. These two implications, together, say that if we form a differential equation

$$\dot{\hat{\mathbf{x}}}(t) = -H\hat{\mathbf{x}}(t) + \mathbb{1}, \tag{2.15}$$

where $t \in [0, \infty)$ denotes continuous time and $\hat{\mathbf{x}}(t) \in \mathbb{R}^{2(N-1)}$ is an estimate of the unknown $\mathbf{x}$ at time t, then for any initial condition $\hat{\mathbf{x}}(0)$, the estimate $\hat{\mathbf{x}}(t)$ would exponentially converge to the unique equilibrium point $\mathbf{x}$ with a convergence rate characterized by the uniform eigenvalues of -1. Therefore, we may define a continuous-time distributed algorithm as follows: for each directed edge $(i, j) \in \tilde{\mathcal{E}}$, let $\hat{x}_{(i,j)}(t) \in \mathbb{R}$ represent an estimate of $x_{(i,j)}$ at time t and suppose $\hat{x}_{(i,j)}(t)$ is maintained in node i's memory. In addition to maintaining $\hat{x}_{(i,j)}(t)$ $\forall j \in \mathcal{N}_i$, suppose each node $i \in \mathcal{V}$ maintains an estimate $\hat{B}_i(t) \in \mathbb{R}$ of B_i and an estimate $\hat{B}_{\{i,j\}}(t) \in \mathbb{R}$ of $B_{\{i,j\}}$ $\forall j \in \mathcal{N}_i$. Equations (2.8), (2.9) and (2.15) collectively suggest the following synchronous and homogeneous rule for updating all the estimates:

$$\dot{\hat{x}}_{(i,j)}(t) = -\hat{x}_{(i,j)}(t) + \sum_{\substack{k \in \mathcal{N}_j \\ k \neq i}} \hat{x}_{(j,k)}(t) + 1, \quad \forall i \in \mathcal{V},\ \forall j \in \mathcal{N}_i, \tag{2.16a}$$

$$\hat{B}_i(t) = \sum_{j \in \mathcal{N}_i} \sum_{\substack{k \in \mathcal{N}_i \\ k \neq j}} \hat{x}_{(i,j)}(t)\hat{x}_{(i,k)}(t), \quad \forall i \in \mathcal{V}, \tag{2.16b}$$

$$\hat{B}_{\{i,j\}}(t) = 2\hat{x}_{(i,j)}(t)\hat{x}_{(j,i)}(t), \quad \forall i \in \mathcal{V},\ \forall j \in \mathcal{N}_i. \tag{2.16c}$$

Notice that (2.16a) is equivalent to (2.15), while (2.16b) and (2.16c) are based on (2.8) and (2.9). Hence, algorithm (2.16) may be viewed as a networked dynamical system with an affine state equation and a nonlinear output equation. Also note that to implement algorithm (2.16), every pair of one-hop neighbors $i, j \in \mathcal{V}$ with $\{i, j\} \in \mathcal{E}$ need to continuously exchange their $\hat{x}_{(i,j)}(t)$ and $\hat{x}_{(j,i)}(t)$. There are, however, no restrictions on the initial condition, no needs to construct shortest paths, and no

algorithm parameters to tune.

The following theorem summarizes the above findings:

Theorem 1. *The continuous-time algorithm* (2.16) *has a unique equilibrium point* $\mathbf{x}$ *that is exponentially stable, such that for any* $\hat{x}_{(i,j)}(0)\forall(i,j)\in\tilde{\mathcal{E}}$*, we have* $\lim_{t\to\infty}\hat{x}_{(i,j)}(t) = x_{(i,j)}\ \forall(i,j)\in\tilde{\mathcal{E}}$*. In addition,* $\lim_{t\to\infty}\hat{B}_i(t) = B_i\ \forall i\in\mathcal{V}$ *and* $\lim_{t\to\infty}\hat{B}_{\{i,j\}}(t) = B_{\{i,j\}}\ \forall\{i,j\}\in\mathcal{E}$*.*

2.4.2 Discrete-Time Algorithm

The continuous-time algorithm (2.16) is made possible by the unipotent property of H established in Lemma 1. In what follows, we make use of the nilpotent property of $I-H$ from Corollary 1 to design its discrete-time counterpart. To this end, notice that if we form a difference equation

$$\hat{\mathbf{x}}(t+1) = (I-H)\hat{\mathbf{x}}(t) + \mathbb{1}, \tag{2.17}$$

where $t\in\{0,1,2,\ldots\}$ here denotes discrete time and $\hat{\mathbf{x}}(t)\in\mathbb{R}^{2(N-1)}$ plays the same role as before, then because of (2.13) and because all the eigenvalues of $I-H$ are 0, $\hat{\mathbf{x}}(t)$ would converge to the unique equilibrium point $\mathbf{x}$ in finite time irrespective of the initial condition $\hat{\mathbf{x}}(0)$. Hence, (2.17) suggests the following discrete-time distributed algorithm for iterating the estimates $\hat{x}_{(i,j)}(t)$'s:

$$\hat{x}_{(i,j)}(t+1) = \sum_{\substack{k\in\mathcal{N}_j\\k\neq i}}\hat{x}_{(j,k)}(t)+1,\quad \forall i\in\mathcal{V},\ \forall j\in\mathcal{N}_i. \tag{2.18}$$

Note that algorithm (2.18) is an affine state equation, and its nonlinear output equations are identical to (2.16b) and (2.16c), except that t here is integer-valued. To prove the convergence of the algorithm (2.18), recall the definition of $\mathcal{V}_{(i,j)}$ in (2.2). Let $\tau_{(i,j)} = \max_{k\in\mathcal{V}_{(i,j)}} d_{ik}$, where d_{ik} is the distance between nodes i and k. We show

below that $\forall \hat{\mathbf{x}}(0)$, $\forall (i,j) \in \tilde{\mathcal{E}}$, $\forall t \in \{\tau_{(i,j)}, \tau_{(i,j)}+1, \ldots\}$, $\hat{x}_{(i,j)}(t) = x_{(i,j)}$. In other words, if we let $\tilde{\mathcal{E}}_\ell = \{(i,j) \in \tilde{\mathcal{E}} : \tau_{(i,j)} = \ell\}$ for $\ell \in \{1, 2, \ldots, D(G)\}$, we have:

Claim 1. *For any arbitrary* $\hat{\mathbf{x}}(0) \in \mathbb{R}^{2(N-1)}$, $\forall \ell \in \{1, 2, \ldots, D(G)\}$, *we have*

$$\hat{x}_{(i,j)}(t) = x_{(i,j)}, \quad \forall (i,j) \in \bigcup_{i=1}^{\ell} \tilde{\mathcal{E}}_i, \forall t \in \{\ell, \ell+1, \ldots\}. \tag{2.19}$$

Proof. Let $\hat{\mathbf{x}}(0)$ be given. We show by induction that (2.19) is true$\forall \ell \in \{1, 2, \ldots, D(G)\}$.

First, let $\ell = 1$ and let $(i,j) \in \tilde{\mathcal{E}}_1$, indicating that node j is a leaf node. Therefore, according to (2.18) $\forall t \in \{1, 2, \ldots\}$, $\hat{x}_{(i,j)}(t) = 1 = x_{(i,j)}$.

Next, suppose (2.19) is true for $\ell \in \{1, 2, \ldots, D(G)-1\}$. Let $(i,j) \in \bigcup_{i=1}^{\ell+1} \tilde{\mathcal{E}}_i$, there are two cases:

- when $(i,j) \in \bigcup_{i=1}^{\ell} \tilde{\mathcal{E}}_i$, based on the proposition, $\forall t \in \{\ell+1, \ell+2, \ldots\}$, we have $\hat{x}_{(i,j)}(t) = x_{(i,j)}$;
- when $(i,j) \in \tilde{\mathcal{E}}_{\ell+1}$, $\forall k \in \mathcal{N}_j / \{i\}$, if $\forall (j,k) \in \bigcup_{i=1}^{\ell} \tilde{\mathcal{E}}_i$, we have $\hat{x}_{(j,k)}(t-1) = x_{(j,k)}$. Thus $\hat{x}_{(i,j)}(t) = x_{(i,j)}$ based on (2.18); and if $\exists$ at least one $(j,k) \in \bigcup_{i=\ell+1}^{D(G)} \tilde{\mathcal{E}}_i$, based on the property of shortest path, $\tau_{(i,j)} = \max_{k \in \mathcal{V}_{(i,j)}} d_{ik} = \max_{k \in \mathcal{V}_{(i,j)}} (d_{ij} + d_{jk}) = 1 + \max_{k \in \mathcal{V}_{(i,j)}} d_{jk} \geq \ell + 2$, which contradicts the fact that $(i,j) \in \tilde{\mathcal{E}}_{\ell+1}$.

□

Corollary 2. *For any arbitrary* $(i,j) \in \tilde{\mathcal{E}}$, $\forall t \in \{D(G), D(G)+1, \ldots\}$, $\hat{x}_{(i,j)}(t) = x_{(i,j)}$.

Proof. By definition, $D(G) = \max_{i \in \mathcal{V}, j \in \mathcal{V}} d_{ij}$. Since $\mathcal{V}$ is a finite set, let $U_D = \{\{i_0, j_0\} : d_{i_0 j_0} = D(G), \forall i_0 \in \mathcal{V}, \forall j_0 \in \mathcal{V}\}$ be the set of pairs of nodes with each pair has distance $D(G)$. $\forall (i,j) \in \tilde{\mathcal{E}}$, $\tau_{(i,j)} = \max_{k \in \mathcal{V}_{(i,j)}} d_{ik} \leq \max_{k \in \mathcal{V}} d_{ik} \leq \max_{i \in \mathcal{V}, k \in \mathcal{V}} d_{ik} = D(G)$, where the equality holds only when $\exists k \in \mathcal{V}_{(i,j)}$ s.t., $\{i,k\} \in U_D$. Therefore, Corollary 2 is true based on (2.19). □

Based on Claim 1 and Corollary 2, as is asserted in the theorem below, algorithm (2.18) not only achieves a deadbeat response, it does so in $t = D(\mathcal{G})$ time steps:

Theorem 2. *The discrete-time algorithm* (2.18) *has a unique equilibrium point* $\mathbf{x}$ *that is finite-time stable, such that for any* $\hat{x}_{(i,j)}(0)\ \forall (i,j) \in \tilde{\mathcal{E}}$ *and for any* $t \in \{D(\mathcal{G}), D(\mathcal{G})+1, D(\mathcal{G})+2, \ldots\}$, *we have* $\hat{x}_{(i,j)}(t) = x_{(i,j)}\ \forall (i,j) \in \tilde{\mathcal{E}}$, $\hat{B}_i(t) = B_i$ $\forall i \in \mathcal{V}$, *and* $\hat{B}_{\{i,j\}}(t) = B_{\{i,j\}}\ \forall \{i,j\} \in \mathcal{E}$.

Proof. The proof is an immediate consequence of Claim 1 and Corollary 2. □

2.4.3 Alternative Discrete-Time Algorithm

This algorithm in (2.18) not only enables nodes in a large-scale network to cooperatively learn how important or critical they individually are, but also to learn how large the network is, in terms of the network size N, with only local interaction and without any global coordination nor knowledge of the network topology. However, in some applications, the network size N is already known by each node. Is it possible to improve the efficiency based on the knowledge of N? This subsection is devoted to addressing this question by introducing an alternative discrete-time algorithm.

Aimed at this goal, for each edge $\{i,j\} \in \mathcal{E}$, let $b_{\{i,j\}} \in \{0,1\}$, $\forall t \in \{0,1,2,\ldots\}$. Suppose each node $i \in \mathcal{V}$ initializes $b_{\{i,j\}}(0)\ \forall j \in \mathcal{N}_i$ to zero. Let $b_{\{i,j\}}(t)$ be defined as

$$b_{\{i,j\}}(t+1) = \begin{cases} 1, & \text{if } b_{\{i,h\}}(t) = 1\ \forall h \in \mathcal{N}_i/\{j\} \text{ or } b_{\{j,k\}}(t) = 1\ \forall k \in \mathcal{N}_j/\{i\}, \\ 0, & \text{otherwise.} \end{cases} \tag{2.20}$$

Note that when $\mathcal{N}_i = \{j\}$ or $\mathcal{N}_j = \{i\}$ at time t, $b_{\{i,j\}}(t+1) = 1$ is true vacuously. Since when node i or j is a leaf node, which is under this condition, we have $\hat{x}_{(j,i)}(t+1) = 1 = x_{(j,i)}$ or $\hat{x}_{(i,j)}(t+1) = 1 = x_{(i,j)}$ respectively, and since if $\hat{x}_{\{i,h\}}(t) = x_{\{i,h\}}$

$\forall h \in \mathcal{N}_i/\{j\}$ or $\hat{x}_{\{j,k\}}(t) = x_{\{j,k\}}$ $\forall k \in \mathcal{N}_j/\{i\}$, we get $\hat{x}_{(i,j)}(t+1) = x_{(i,j)}$ according to (2.10) and (2.12). Therefore, $b_{\{i,j\}}(t+1)$ in (2.20) represents the flag of whether $\hat{x}_{(i,j)}(t+1) = x_{(i,j)}$ is achieved.

Recall from Section 2.3 that if each node $i \in \mathcal{V}$ is able to determine the values of $x_{(i,j)}$ and $x_{(j,i)}$ $\forall j \in \mathcal{N}_i$, then it could use (2.8) and (2.9) to compute B_i and $B_{\{i,j\}}$ $\forall j \in \mathcal{N}_i$ by itself. In light of this fact, $\forall t \in \{0, 1, 2, \ldots\}$, for each directed edge $(i,j) \in \tilde{\mathcal{E}}$, let $\hat{x}_{(i,j)}(t) \in \mathbb{R}$ and $b_{\{i,j\}}(t) \in \mathbb{R}$ are maintained in node i's memory (i.e., each node $i \in \mathcal{V}$ maintains $\hat{x}_{(i,j)}(t)$ and $b_{\{i,j\}}(t)$ $\forall j \in \mathcal{N}_i$). In addition, for each $i \in \mathcal{V}$, $\hat{B}_i(t)$ and $\hat{B}_{\{i,j\}}(t)$ $\forall j \in \mathcal{N}_i$ are also maintained in node i's memory. Suppose each node $i \in \mathcal{V}$ initializes $\hat{x}_{(i,j)}(0)$ $\forall j \in \mathcal{N}_i$ arbitrarily. Based on (2.10) and (2.12), $\forall i \in \mathcal{V}$, $\forall j \in \mathcal{N}_i$, for each $t \in \{0, 1, 2, \ldots\}$, along with (2.20), we obtain another rule for updating all the estimates:

$$\hat{x}_{(i,j)}(t+1) = \begin{cases} N - (1 + \sum\limits_{\substack{h \in \mathcal{N}_i \\ h \neq j}} \hat{x}_{(i,h)}(t)), & \text{if } b_{\{i,h\}}(t) = 1 \ \forall h \in \mathcal{N}_i/\{j\}, \\ 1 + \sum\limits_{\substack{k \in \mathcal{N}_j \\ k \neq i}} \hat{x}_{(j,k)}(t), & \text{otherwise.} \end{cases} \tag{2.21}$$

Similar to algorithm (2.18), algorithm (2.20) and (2.21) consists of two state equations, and its nonlinear output equations are identical to (2.16b) and (2.16c) with discrete time t here. Moreover, as is asserted in the theorem below, with the knowledge of N, algorithm (2.20) and (2.21) further reduce the finite convergence time to $t = \lceil \frac{D(\mathcal{G})}{2} \rceil$ time steps:

Theorem 3. *The discrete-time algorithm* (2.20) *and* (2.21) *has a unique equilibrium point* $\mathbf{x}$ *that is finite-time stable, such that for any* $\hat{x}_{(i,j)}(0)$ $\forall (i,j) \in \tilde{\mathcal{E}}$ *and for any* $t \in \{\lceil \frac{D(\mathcal{G})}{2} \rceil, \lceil \frac{D(\mathcal{G})}{2} \rceil + 1, \lceil \frac{D(\mathcal{G})}{2} \rceil + 2, \ldots\}$, *we have* $\hat{x}_{(i,j)}(t) = x_{(i,j)}$ $\forall (i,j) \in \tilde{\mathcal{E}}$, $\hat{B}_i(t) = B_i$ $\forall i \in \mathcal{V}$, *and* $\hat{B}_{\{i,j\}}(t) = B_{\{i,j\}}$ $\forall \{i,j\} \in \mathcal{E}$.

2.4.4 Asynchronous Distributed Algorithm

The aforementioned discrete-time distributed algorithms are relatively simple to design and always provide a known upper bound on the process execution. However, as the size of network becomes very large, this synchronization may become difficult. Thus, it is desirable to design an asynchronous distributed algorithm, which allows variables be updated independently and without the need for synchronization.

To begin, let t denote the index of iterations. Suppose $\forall t \in \{0, 1, 2, \ldots\}$, each node $i \in \mathcal{V}$ maintains in its local memory an estimate $\hat{x}_{(i,j)}(t)$ of $x_{(i,j)}$ $\forall j \in \mathcal{N}_i$, with an arbitrary initial conditions, i.e., $\hat{x}_{(i,j)}(0) \in \mathbb{R}$ $\forall j \in \mathcal{N}_i$. Suppose at each iteration $t \in \{0, 1, 2, \ldots\}$, a node $u \in \mathcal{V}$ is selected to initiate the iteration with a selected neighbor $v \in \mathcal{N}_u$. During this process, node u updates its variable $\hat{x}_{(u,v)}(t)$, whereas the rest of the variables remain idle. To determine how these variables are updated, we introduce below an objective function V, defined as

$$V(\hat{\mathbf{x}}(t)) = \|H\hat{\mathbf{x}}(t) - \mathbb{1}\|^2. \tag{2.22}$$

Due to (2.13), we have

$$V(\mathbf{x}) = 0$$

Since H is nonsingular,

$$V(\hat{\mathbf{x}}) > 0 \quad \text{if } \hat{\mathbf{x}} \neq \mathbf{x}.$$

Thus, V is quadratic positive definite with respect to $\mathbf{x}$, making it a legitimate Lyapunov function candidate.

Thus, by having the sequence of all pairs of nodes repeatedly updating their estimates so that the value of $V(\hat{\mathbf{x}}(t))$ is repeatedly minimized, all the N nodes in the

network would gradually drive their estimates toward the true values, thereby driving their estimates $\hat{B}_i(t)$'s and $\hat{B}_{\{i,j\}}(t)$'s toward the true B_i's and $B_{\{i,j\}}$'s.

Note that V in (2.22) can be written using (2.12) as its elementary form:

$$V(\hat{\mathbf{x}}(t)) = \sum_{(i,j)\in\tilde{\mathcal{E}}} \Big(\hat{x}_{(i,j)}(t) - \sum_{\substack{k\in\mathcal{N}_j \\ k\neq i}} \hat{x}_{(j,k)}(t) - 1\Big)^2. \tag{2.23}$$

Suppose $\hat{x}_{(u,v)}(t+1)$ is chosen to minimize $V(\hat{\mathbf{x}}(t+1)) - V(\hat{\mathbf{x}}(t))$, which equals to minimize $V(\hat{\mathbf{x}}(t+1))$ since $V(\hat{\mathbf{x}}(t))$ can be viewed as a constant at $t+1$. let

$$\hat{\mathbf{x}}(t) = \Big[\hat{x}_{(i_1,j_1)}(t), \ldots, \hat{x}_{(u,v)}(t), \ldots, \hat{x}_{(i_{2(N-1)},j_{2(N-1)})}(t)\Big]^T$$

be the estimate of $\mathbf{x}$ at t, i.e.,

$$\hat{x}_{(u,v)}(t+1) = \arg\min_{z\in\mathbb{R}} \|H\left[\hat{x}_{(i_1,j_1)}(t), \ldots, z, \ldots, \hat{x}_{(i_{2(N-1)},j_{2(N-1)})}(t)\right]^T - \mathbb{1}\|^2.$$

To calculate its value, take the partial derivative of $V(\hat{\mathbf{x}}(t+1))$ with respect to $\hat{x}_{(u,v)}(t+1)$ and set it to zero, yields

$$\hat{x}_{(u,v)}(t+1) = \frac{1}{|\mathcal{N}_u|}\Big[\sum_{\substack{k\in\mathcal{N}_v \\ k\neq u}} \hat{x}_{(v,k)}(t) + \sum_{\substack{k\in\mathcal{N}_u \\ k\neq v}} \hat{x}_{(k,u)}(t) - (|\mathcal{N}_u| - 2)\sum_{\substack{k\in\mathcal{N}_u \\ k\neq v}} \hat{x}_{(u,k)}(t) + 2 - |\mathcal{N}_u|\Big].$$

In other words, $\forall (i,j) \in \tilde{\mathcal{E}}$, $\hat{x}_{(i,j)}(t+1) = \hat{x}_{(u,v)}(t+1)$ only when $\hat{x}_{(i,j)}$ is the one being updated at $t+1$, otherwise $\hat{x}_{(i,j)}$ will remain idle, i.e.,

$$\hat{x}_{(i,j)}(t+1) = \begin{cases} \frac{1}{|\mathcal{N}_i|}\Big[\sum_{\substack{k\in\mathcal{N}_j \\ k\neq i}} \hat{x}_{(j,k)}(t) + \sum_{\substack{k\in\mathcal{N}_i \\ k\neq j}} \hat{x}_{(k,i)}(t) - (|\mathcal{N}_i| - 2)\sum_{\substack{k\in\mathcal{N}_i \\ k\neq j}} \hat{x}_{(i,k)}(t) + 2 - |\mathcal{N}_i|\Big], \\ \qquad\qquad \text{if } \hat{x}_{(i,j)} \text{ is being updated at } t+1, \\ \hat{x}_{(i,j)}(t), \qquad\qquad \text{otherwise,} \end{cases} \tag{2.24}$$

along with (2.16b) and (2.16c), providing the asynchronous distributed algorithm. To analyze its stability and for the sake of convenience, let's first consider a linear system of equations

$$Az = b$$

where $A \in \mathbb{R}^{n\times n}$, $z \in \mathbb{R}^n$, $b \in \mathbb{R}^n$, A is nonsingular and $A = [a_1, \ldots, a_n]$, $a_i \in \mathbb{R}^n$, which is an simplified version of our problem in (2.13), where

$$A = H,$$
$$z = \mathbf{x},$$
$$b = \mathbb{1},$$

and $n = 2(N-1)$. Let $t \in \{0, 1, 2, \ldots\}$ denote the index of iterations, and let

$$z(t) = \Big[z_1(t), z_2(t), \ldots, z_n(t)\Big]^T$$

be the estimate of z^* at t, $\forall t \in \{0, 1, 2, \ldots\}$. Consider the following update rule:

$$z_i(t+1) = \begin{cases} \arg\min_{z\in\mathbb{R}} \|A \Big[z_1(t), \ldots, z_{i-1}(t), z, z_{i+1}(t), \ldots, z_n(t)\Big]^T - b\|^2, \\ \qquad\qquad\qquad\qquad \text{if } z_i \text{ is being updated at } t+1, \\ z_i(t), \qquad\qquad\qquad\qquad \text{otherwise,} \end{cases} \tag{2.25}$$

In this case, we can think of the variable $\hat{x}_{(i,j)}(t+1)$, as $z_i(t+1)$ in (2.25). Moreover, let z^* be the unique solution to (2.25), i.e., $Az^* = b$. We show from lemma below, the asynchronous algorithm (2.25) is exponentially stable:

Lemma 2. *For each $t \in \{0, 1, 2, \ldots\}$, let $u(t) \in \{1, 2, \ldots, n\}$ be the variable that gets*

updated. For each $i \in \{0, 1, 2, \ldots, n\}$*, let* $U_i = \{t \in \{0, 1, 2, \ldots\} : u(t) = i\}$ *be the index of iterations that* i *gets updated. Assume that for each* $i \in \{0, 1, 2, \ldots, n\}$*, the set* U_i *has infinitely many elements and that* $\exists T < \infty$ *s.t. for each* $t \in \{0, 1, 2, \ldots\}$*,* $\exists l \in U_i$ *s.t.* $l \in \{t + 1, t + 2, \ldots, t + T\}$*. The asynchronous algorithm* (2.25) *has a unique equilibrium point* z^* *that is exponentially stable.*

Proof. Consider, as in our development, a Lyapunov function candidate V:$\mathbb{R}^n \to \mathbb{R}$, defined as

$$V(z(t)) = \|Az(t) - b\|^2, \quad \forall t \in \{0, 1, 2, \ldots\}.$$

Clearly,

$$V(z^*) = \|Az^* - b\|^2 = 0,$$

and

$$V(z) > 0 \quad \forall z \neq z^*$$

due to A being nonsingular. Thus, V is positive definite with respect to z^*. Due to (2.25), the change in the value of V as we go from t to $t + 1$ is

$$V(z(t + 1)) - V(z(t)) = -\|a_i\|^2 \Big(z_i(t + 1) - z_i(t)\Big)^2, \tag{2.26}$$

where z_i is being updated at $t + 1$. For convenience, let $\tilde{z}(t) = Az(t) - b$, so that

$$\tilde{z}(t + 1) = P_i \tilde{z}(t), \quad \text{if } i \text{ updates at } t + 1 \tag{2.27}$$

$$V(t) = \|\tilde{z}(t)\|^2, \tag{2.28}$$

where

$$P_i = I - \frac{1}{\|a_i\|^2} a_i a_i^T.$$

Moreover, (2.26) can be rewritten as

$$V(t) - V(t+1) = \|\tilde{z}(t+1) - \tilde{z}(t)\|^2 \geq 0, \tag{2.29}$$

implying that the sequence $V(0), V(1), V(2), \ldots$ is a non-increasing, nonnegative, and therefore convergent sequence.

From the assumption and (2.27), for each $t_0 \in \{0, 1, 2, \ldots\}$, $\forall i \in \{0, 1, 2, \ldots, n\}$, $\exists l \in \{t_0 + 1, t_0 + 2, \ldots, t_0 + T\}$ s.t.,

$$\tilde{z}(l) = P_i \tilde{z}(l-1).$$

Thus,

$$\begin{aligned} a_i^T \tilde{z}(l) &= a_i^T P_i \tilde{z}(l-1) \\ &= a_i^T [I - \frac{1}{\|a_i\|^2} a_i a_i^T] \tilde{z}(l-1) \\ &= a_i^T \tilde{z}(l-1) - a_i^T \tilde{z}(l-1) \\ &= 0. \end{aligned}$$

Then,

$$\begin{aligned}|a_i^T\tilde{z}(t_0+T)|^2 =&|a_i^T\Big(\tilde{z}(t_0+T)-\tilde{z}(l)\Big)|^2\\ \leq&\|a_i\|^2\|\tilde{z}(t_0+T)-\tilde{z}(l)\|^2\\ =&\|a_i\|^2\|\tilde{z}(t_0+T)-\tilde{z}(t_0+T-1)+\tilde{z}(t_0+T-1)\\ &-\tilde{z}(t_0+T-2)+\cdots+\tilde{z}(l+1)-\tilde{z}(l)\|^2.\end{aligned}$$

Applying the Cauchy-Schwarz Inequality to it,

$$\begin{aligned}|a_i^T\tilde{z}(t_0+T)|^2 \leq&\|a_i\|^2(t_0+T-l)\Big(\|\tilde{z}(t_0+T)-\tilde{z}(t_0+T-1)\|^2+\|\tilde{z}(t_0+T-1)\\ &-\tilde{z}(t_0+T-2)\|^2+\cdots+\|\tilde{z}(l+1)-\tilde{z}(l)\|^2\Big)\\ \leq&\|a_i\|^2(t_0+T-l)\Big(\|\tilde{z}(t_0+T)-\tilde{z}(t_0+T-1)\|^2+\|\tilde{z}(t_0+T-1)\\ &-\tilde{z}(t_0+T-2)\|^2+...+\|\tilde{z}(l+1)-\tilde{z}(l)\|^2+...+\|\tilde{z}(t_0+1)-\tilde{z}(t_0)\|^2\Big).\end{aligned}$$

Due to (2.29), $\forall i \in \{0,1,2,\ldots,n\}$, we obtain

$$\begin{aligned}|a_i^T\tilde{z}(t_0+T)|^2 =&\|a_i\|^2(t_0+T-l)\Big(-V(t_0+T)+V(t_0+T-1)-V(t_0+T-1)+...\\ &-V(t_0+1)+V(t_0)\Big)\\ =&\|a_i\|^2(t_0+T-l)\Big(-V(t_0+T)+V(t_0)\Big)\\ \leq&\|a_i\|^2(T-l)\Big(V(t_0)-V(t_0+T)\Big).\end{aligned}\tag{2.30}$$

Based on (2.28), (2.30), and since $l \geq 1$,

$$\begin{aligned} V(t_0+T) &= \|\tilde{z}(t_0+T)\|^2 \\ &\leq \frac{1}{\lambda_{\min}(AA^T)} \tilde{z}^T(t_0+T) AA^T \tilde{z}(t_0+T) \\ &= \frac{1}{\lambda_{\min}(AA^T)} \sum_{i=1}^{n} |a_i^T \tilde{z}(t_0+T)|^2 \\ &\leq \frac{\sum_{i=1}^{n} \|a_i\|^2 (T-1)(V(t_0)-V(t_0+T))}{\lambda_{\min}(AA^T)}. \end{aligned} \tag{2.31}$$

Note that $\lambda_{\min}(AA^T) = \sigma_{\min}^2(A)$ and $\sum_{i=1}^{n} \|a_i\|^2 = \sum_{i=1}^{n} \sigma_i^2(A)$. Thus, from (2.31), we have

$$V(t_0+T) \leq \gamma V(t_0), \tag{2.32}$$

where

$$\gamma = \frac{(T-1)\sum_{i=1}^{n} \sigma_i^2(A)}{\sigma_n^2(A) + (T-1)\sum_{i=1}^{n} \sigma_i^2(A)}.$$

Next, $\forall k \in \{0,1,2,\ldots\}$, $\forall \alpha \in \{0,1,2,\ldots,T-1\}$, let $t = kT+\alpha$. Based on (2.29) and (2.32),

$$\begin{aligned} V(t) &= V(kT+\alpha) \\ &\leq V(kT) \\ &\leq \gamma^k V(0) \\ &= \frac{\gamma^{t/T}}{\gamma^{\alpha/T}} V(0). \end{aligned}$$

Since $0 < \gamma < 1$ and $0 \leq \alpha/T < 1$, we have

$$\gamma^{\alpha/T} > \gamma,$$

so that

$$V(t) < \frac{V(0)}{\gamma}\gamma^{t/T}, \quad \forall t \in \{0, 1, 2, \ldots\}. \tag{2.33}$$

Next, since $V(z(t)) = \|Az(t) - b\|^2$ and $Az^* = b$,

$$\begin{aligned} V(z(t)) &= (Az(t) - b)^T(Az(t) - b) \\ &= (Az(t) - Az^*)^T(Az(t) - Az^*) \\ &= (z(t) - z^*)^T A^T A(z(t) - z^*). \end{aligned}$$

Since A is nonsingular, $A^T A$ is positive definite with $\lambda_{\min}(A^T A) > 0$. Thus,

$$V(z(t)) \geq \lambda_{\min}(A^T A)\|z(t) - z^*\|^2. \tag{2.34}$$

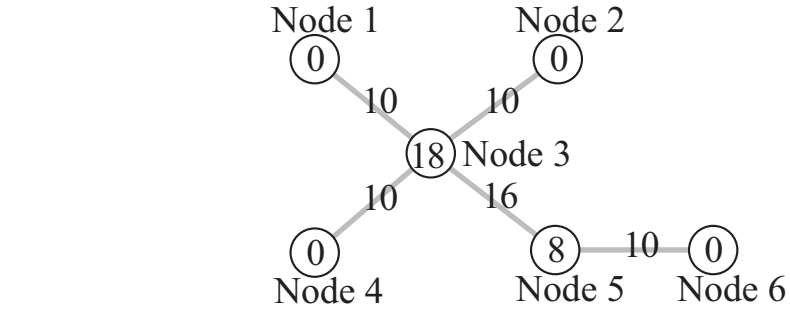
Combining (2.33) and (2.34), we have

$$\|z(t) - z^*\| \leq \sqrt{\frac{V(0)}{\gamma}}\sqrt{\frac{1}{\lambda_{\min}(A^T A)}}\gamma^{\frac{t}{2T}}. \tag{2.35}$$

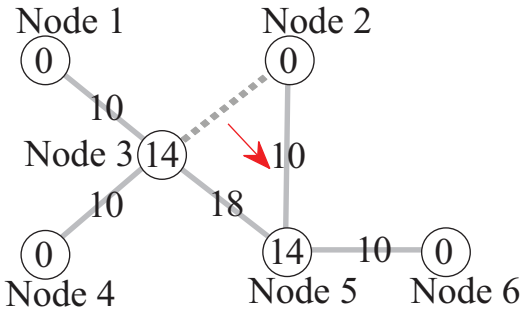
□

Next, if we let $A = H$, $z = \mathbf{x}$, $b = \mathbb{1}$, and $n = 2(N-1)$, as an immediate consequence of Lemma 2, we show in the following theorem that algorithm (2.24) is exponentially stable.

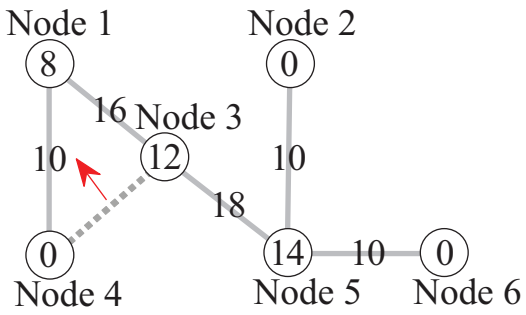
Theorem 4. *The asynchronous distributed algorithm* (2.24) *has a unique equilibrium point* $\mathbf{x}$ *that is exponentially stable, such that for any* $\hat{x}_{(i,j)}(0)$ $\forall (i,j) \in \tilde{\mathcal{E}}$*, we have* $\lim_{t\to\infty} \hat{x}_{(i,j)}(t) = x_{(i,j)}$ $\forall (i,j) \in \tilde{\mathcal{E}}$*. In addition,* $\lim_{t\to\infty} \hat{B}_i(t) = B_i$ $\forall i \in \mathcal{V}$ *and* $\lim_{t\to\infty} \hat{B}_{\{i,j\}}(t) = B_{\{i,j\}}$ $\forall \{i,j\} \in \mathcal{E}$*.*



(a) Over $t \in [0, 10]$.



(b) Over $t \in [10, 20]$.



(c) Over $t \in [20, 30]$.

Figure 2.7: A 6-node tree graph and its node and edge betweenness over time $t \in [0, 30]$.

2.5 Simulation Results

In this section, we present two sets of simulation results that demonstrate the effectiveness of the continuous- and discrete-time synchronous distributed algorithms.

2.5.1 Simulation of the Continuous-Time Distributed Algorithm

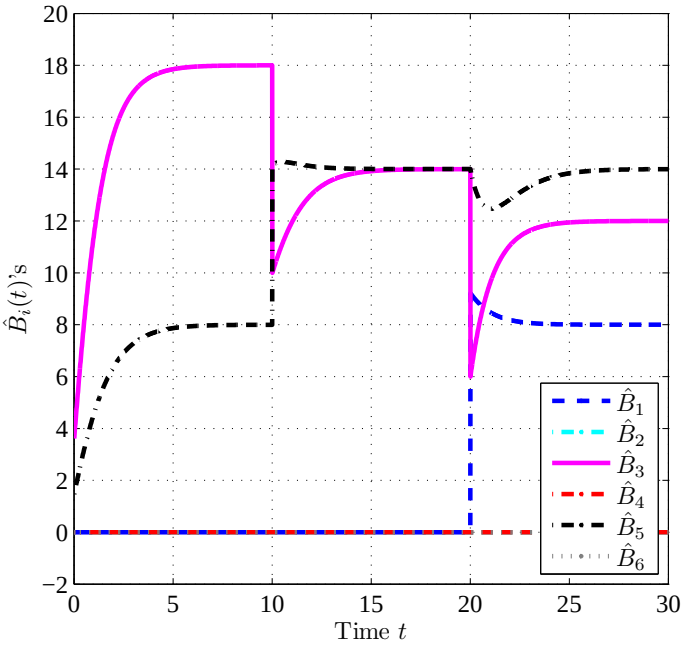
Consider a tree graph with $N = 6$ nodes, whose topology changes from time to time, as shown in Figure 2.7. Specifically, Figure 2.7(a) shows the tree topology over time $t \in [0, 10]$ and the resulting node and edge betweenness B_i's and $B_{\{i,j\}}$'s calculated using (1.1) and (2.1), while Figures 2.7(b) and 2.7(c) do the same for $t \in [10, 20]$ and $t \in [20, 30]$, respectively. As indicated by the arrows and dotted lines in Figures 2.7(b) and 2.7(c), edge $\{2, 3\}$ is deleted and replaced by edge $\{2, 5\}$ at time $t = 10$, while edge

$\{3, 4\}$ is deleted and replaced by edge $\{1, 4\}$ at time $t = 20$, but the graph remains a tree. Suppose the nodes employ the continuous-time algorithm (2.16) to help them cooperatively compute their estimates $\hat{B}_i(t)$'s and $\hat{B}_{\{i,j\}}(t)$'s of the changing B_i's and $B_{\{i,j\}}$'s.

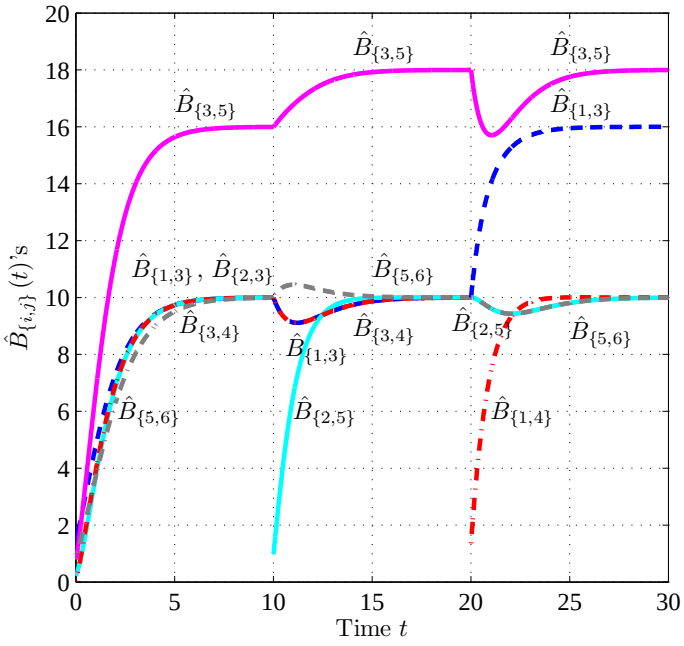
Figure 2.8 displays the simulation result. Observe that despite the time-varying topology, algorithm (2.16) allows the nodes' $\hat{B}_i(t)$'s and $\hat{B}_{\{i,j\}}(t)$'s to asymptotically track the B_i's and $B_{\{i,j\}}$'s without having to "restart" or "refresh"—an action that would likely be required if the algorithm were based on explicit shortest path construction. Indeed, the nodes may not even be aware, and are not required to notify or be notified, that the topology has changed. Also note that because of the time-varying topology, $B_{\{2,3\}}$ is defined only for $t \in [0, 10]$, $B_{\{2,5\}}$ for $t \in [10, 30]$, $B_{\{3,4\}}$ for $t \in [0, 20]$, and $B_{\{1,4\}}$ for $t \in [20, 30]$—and so are their estimates $\hat{B}_{\{i,j\}}(t)$'s.

2.5.2 Simulation of the Discrete-Time Distributed Algorithm

Consider a tree graph with $N = 16$ nodes as shown in Figure 2.9, in which Figure 2.9(a) displays the node indices and Figure 2.9(b) displays the corresponding node and edge betweenness B_i's and $B_{\{i,j\}}$'s. Notice that the diameter of this tree graph is 11 (attained by the shortest path between nodes 7 and 16). Suppose the nodes utilize the discrete-time algorithm (2.18) to jointly compute their estimates $\hat{B}_i(t)$'s and $\hat{B}_{\{i,j\}}(t)$'s of the B_i's and $B_{\{i,j\}}$'s. Figure 2.10 shows the simulation result, where it can be seen that algorithm (2.18) indeed exhibits a deadbeat response, allowing all the nodes' $\hat{B}_i(t)$'s and $\hat{B}_{\{i,j\}}(t)$'s to reach the B_i's and $B_{\{i,j\}}$'s in finite time of no more than $t = 11$ time steps.

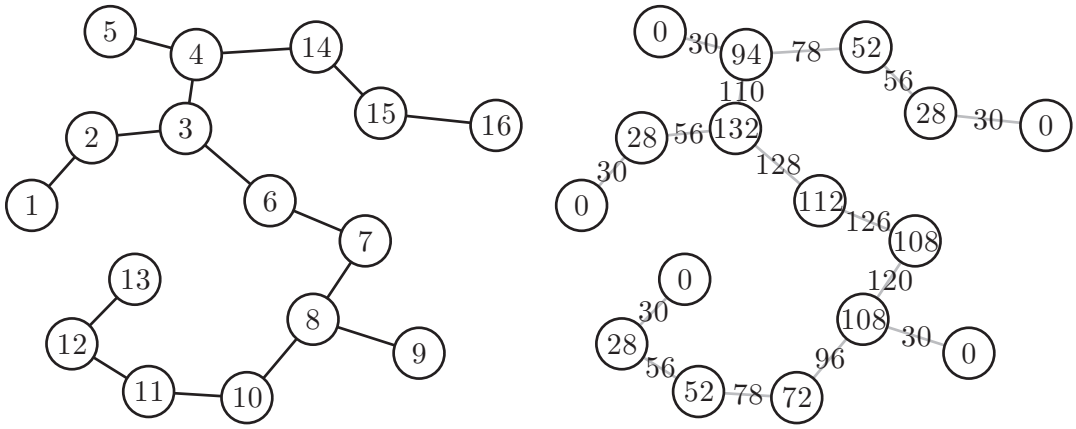


(a) Node betweenness estimates.



(b) Edge betweenness estimates.

Figure 2.8: Performance of the continuous-time algorithm (2.16) in computing node and edge betweenness on the time-varying 6-node tree graph.



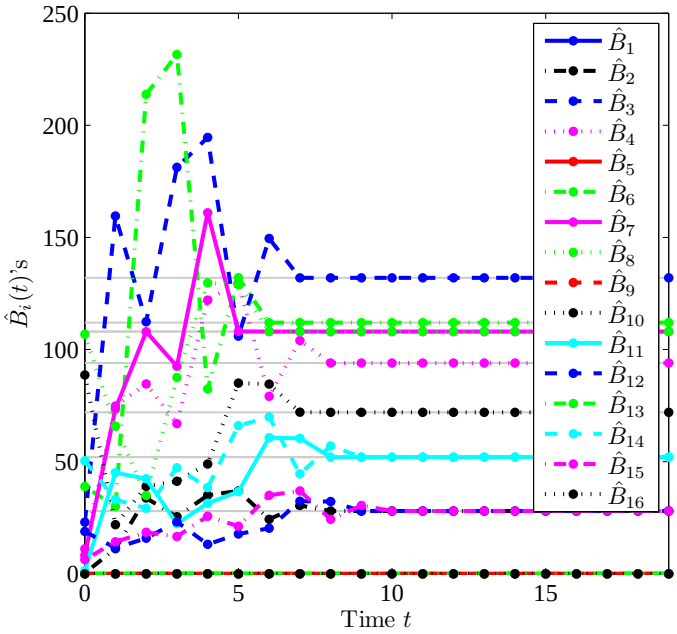
(a) Node indices. (b) Node and edge betweenness.

Figure 2.9: A 16-node tree graph and its node indices and node and edge betweenness.

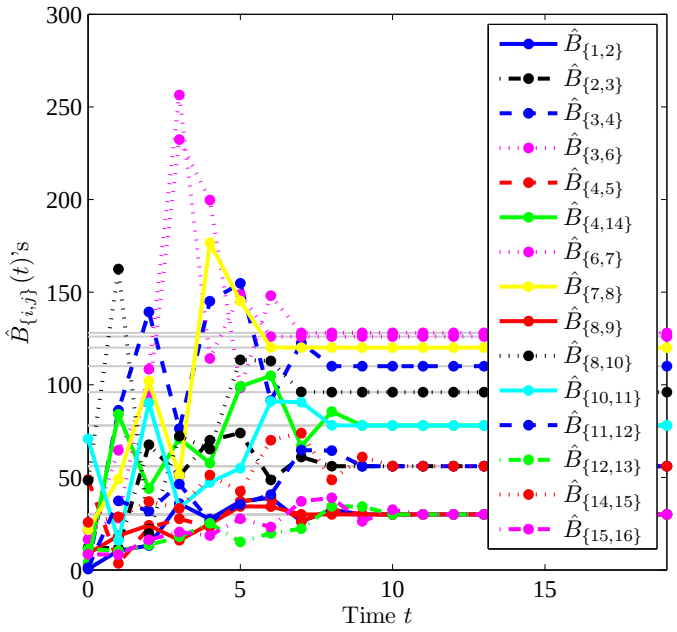
2.6 Conclusion

In this chapter, we have introduced a set of dynamical systems approaches to develop several synchronous (continuous- and discrete-time) and asynchronous distributed algorithms, which enable enable every node in an undirected and unweighted tree graph to cooperatively compute their individual node betweenness and incident edge betweenness with only local interaction and without any centralized coordination. Constructed using dynamical systems approaches, we have shown that the algorithms are simple and scalable, with the synchronous continuous-time algorithm being unconditionally exponentially convergent, the synchronous discrete-time algorithm unconditionally exhibiting a deadbeat response, and the asynchronous algorithm being asymptotically stable. Moreover, we have shown that they require minimal node memories to execute, bypass entirely the need to construct shortest paths.

Given that trees are a very special type of graphs, Chapter 4 will develop distributed algorithms for estimating betweenness on general graphs, building on the ideas of this chapter.



(a) Node betweenness estimates.



(b) Edge betweenness estimates.

Figure 2.10: Performance of the discrete-time algorithm (2.18) in computing node and edge betweenness on the 16-node tree graph.

Chapter 3

Distributed Computation of Closeness on Tree Graphs

3.1 Introduction

When humans analyze a network, it is often useful for them to have measures that assign a score to each node (or edge) in the network, so that meaningful observations can be made about, say, the node importance. Recognizing this need, a growing set of measures known as *centrality measures* has been introduced in the network science literature (e.g., [6, 7, 13, 112–116]). For example, *betweenness centrality* [6, 7], which assigns higher scores to nodes who lie on higher percentages of the shortest paths between all pairs of nodes, and *closeness centrality* [13], which assigns higher scores to those who have shorter distances to all other nodes, are two of the oldest and most fundamental measures that have been widely used.

With the continued increase in the size and complexity of networks and the rising demand for their autonomy, it is becoming increasingly desirable that nodes in a network can analyze the network themselves, as opposed to relying on humans. In particular, it would be useful if the nodes could cooperatively determine their own

betweenness and closeness centralities, despite knowing only who their neighbors are and having limited computational powers, limited physical memories, and no centralized coordination.

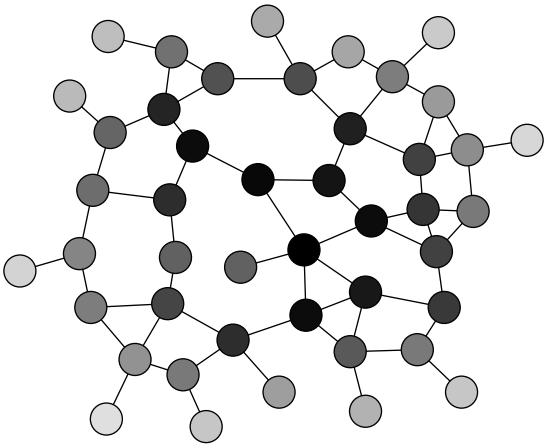
In the previous chapter, we develop a pair of distributed algorithms that enable nodes in a tree graph to compute two variants of betweenness centrality called *node betweenness* and *edge betweenness* [6, 7], which assigns higher scores to nodes who lie on higher percentages of the shortest paths between all pairs of nodes, despite facing the aforementioned constraints. Following the same footprint, in this chapter we address the distributed computation of two variants of closeness centrality called *classic closeness* [12, 13] and *exponential closeness* [113], which assigns higher scores to those who have shorter distances to all other nodes, are two of the oldest and most fundamental measures that have been widely used.

To set the stage, we define these two variants below in the context of an undirected, unweighted and connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, \ldots, N\}$ denotes the set of $N \geq 2$ nodes and $\mathcal{E} \subset \{\{i, j\} : i, j \in \mathcal{V}, i \neq j\}$ denotes the set of edges:

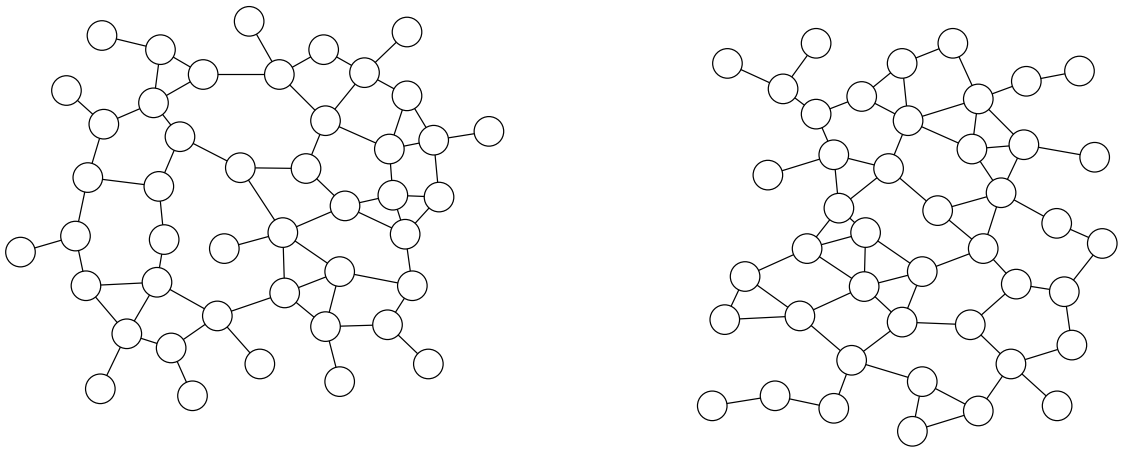
- The *classic closeness* (often referred to simply as *closeness*) C_i of a node $i \in \mathcal{V}$ is defined in Bavelas [12] and Sabidussi [13] in (1.2) as

$$C_i \triangleq \frac{N-1}{\sum_{j \in \mathcal{V}} d_{ij}},$$

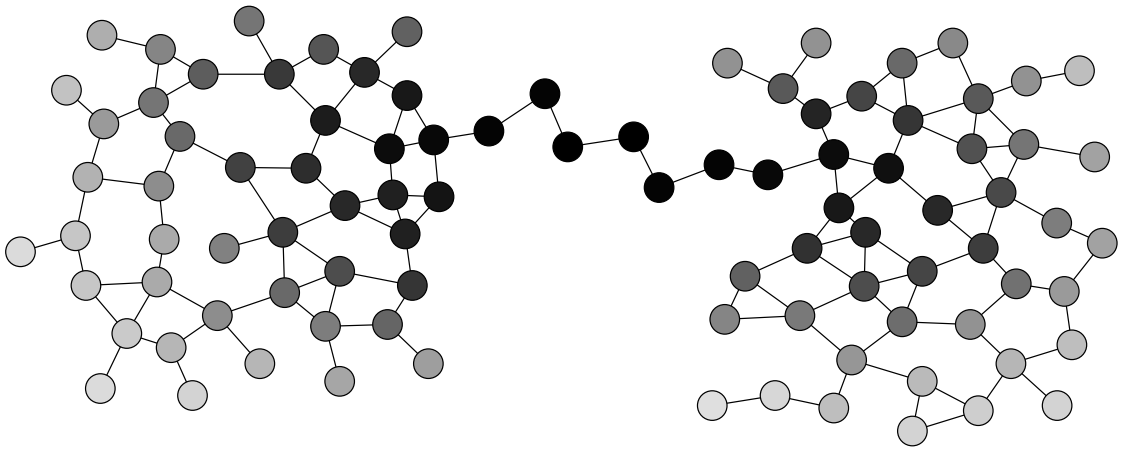
  where $d_{ij} = d_{ji}$ is the distance between nodes i and j, and the factor $N - 1$ is inserted so that $C_i \in (0, 1]$. It follows that the larger C_i, the closer node i is, on average, to all other nodes in the graph $\mathcal{G}$. This makes nodes with high closeness effective in, for instance, spreading diseases or rumors to, and gathering information from, everyone else. Figure 3.1 illustrates this concept, in which the larger C_i, the darker node i is shaded. Notice that classic closeness changes rather gracefully from one node to its neighbors.



(a) Original graph.

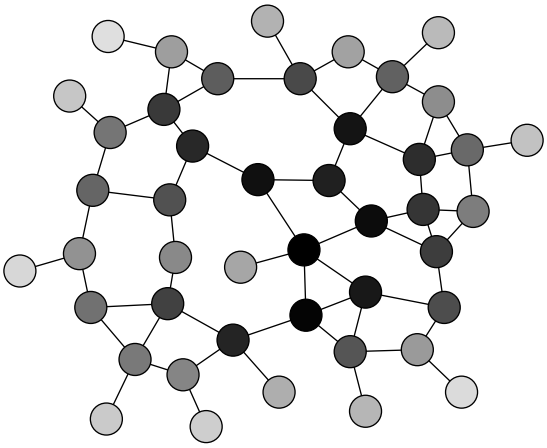


(b) Disconnected graph.

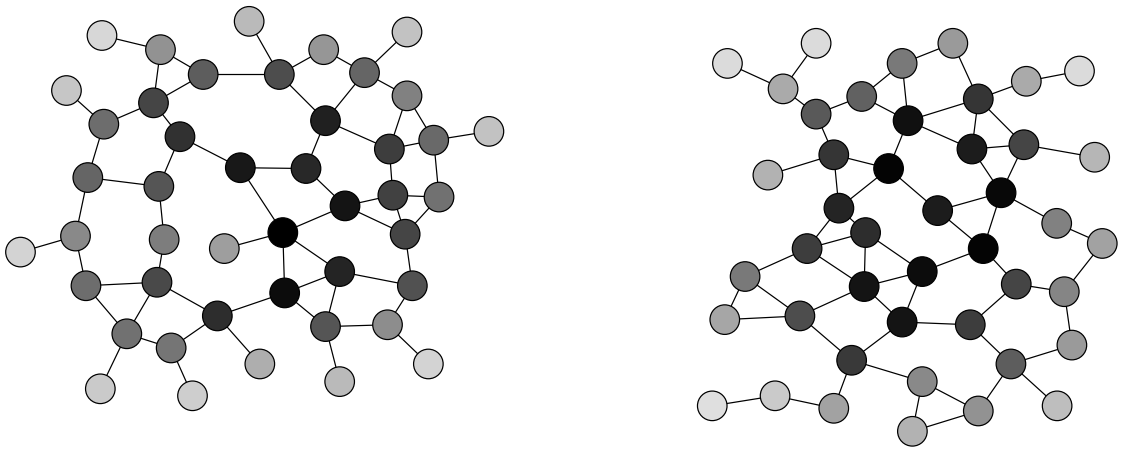


(c) Adding a bridge.

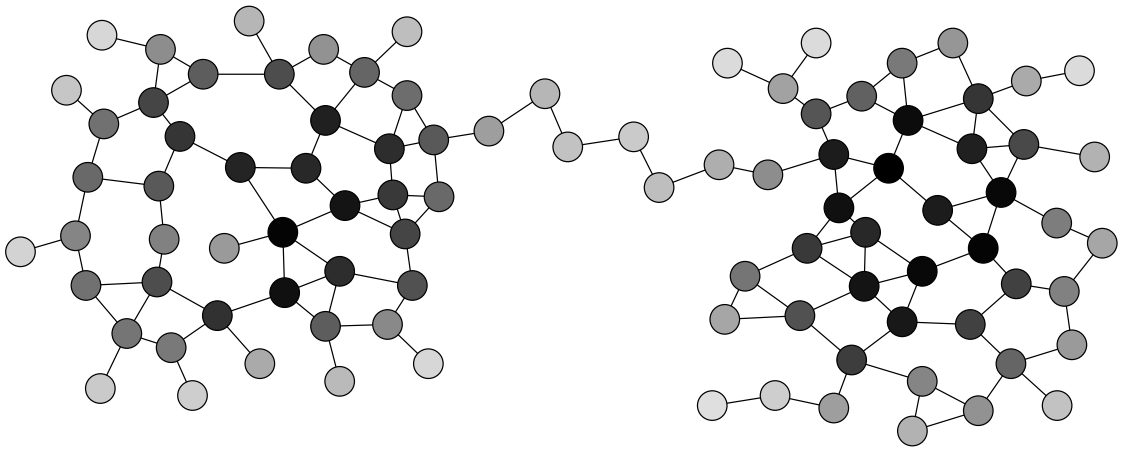
Figure 3.1: An illustration of classic closeness.



(a) Original graph.



(b) Disconnected graph.



(c) Adding a bridge.

Figure 3.2: An illustration of exponential closeness.

- A limitation of classic closeness C_i in (1.2) is that if node i is very far away from some node j, then even if node i is very close to the rest of the nodes, its C_i would be practically zero, signifying that node i has poor closeness. In some applications, it might be desirable to discount the influence of nodes who are very far away, preventing them from "skewing" the closeness of node i. The *exponential closeness* C_i^E of a node $i \in \mathcal{V}$, defined in [113] as

$$C_i^E \triangleq \sum_{\substack{j \in \mathcal{V} \\ j \neq i}} 2^{-d_{ij}},$$

is a measure that possesses this desirable property. In this chapter, we slightly generalize the definition of C_i^E to

$$C_i^E \triangleq \sum_{\substack{j \in \mathcal{V} \\ j \neq i}} \alpha^{-d_{ij}}, \tag{3.1}$$

where $\alpha > 1$, allowing the base of the exponent to be a number other than 2. Figure 3.2 illustrates the idea of exponential closeness and, together with Figure 3.1, demonstrates the difference between C_i and C_i^E:

  - First, from Figure 3.1(a) and 3.2(a) we see that the nodes who are important respected to C_i, are also important respected to C_i^E.
  - However, from Figure 3.1(b) and 3.2(b) we can see that C_i is not well defined for disconnected graph, while C_i^E is.
  - Finally, if you add another component to the graph and connect these two components by a bridge, the two measures are quite different, which can be seen from Figure 3.1(c) and 3.2(c).

The Figure 3.1 and Figure 3.2 show that the classic closeness indeed care about the average distance, so that the most "central" node is the one that lies in the

center of the whole network, i.e., the bridge. In contrast, exponential closeness doesn't care about nodes who are very far away, so that the most "central" nodes are the ones that lies in the center of their own component.

In this chapter, we show that if the graph $\mathcal{G}$ is a *tree*, it is possible to construct continuous- and discrete-time distributed algorithms, which enable every node $i \in \mathcal{V}$ in the tree $\mathcal{G}$ to compute its own classic closeness C_i and exponential closeness C_i^E by talking only to neighbors, executing simple homogeneous update rules, and consuming minimal physical memories. To construct these algorithms, we adopt what we call an *algebraic-relationships-turned-dynamical-systems* approach that was also used in Chapter 2 for computing betweenness. With this approach, we first express the unknown C_i's and C_i^E's as functions of some variables that have graph-theoretic meaning. We then relate these variables to arrive at a set of algebraic equations. By turning these algebraic equations into state equations, and those functions into output equations, we subsequently obtain networked dynamical systems describing the algorithms. We show that the state equation of each system is affine and has a unique equilibrium point that is always exponentially stable (for continuous-time) or finite-time stable (for discrete-time), regardless of the tree topology. Moreover, the output equation, when evaluated at the equilibrium point, always yields the unknown C_i's or C_i^E's, thereby solving the problem.

To our understanding, [23] is the only prior work on distributed computation of betweenness and closeness centralities. The algorithms in [23], however, are notably different from the ones here and in Chapter 2. Specifically, each algorithm in [23] is applicable to general graphs but is memory intensive (i.e., every node has to store a list of messages), non-homogeneous (i.e., nodes act differently depending on what they have received and which phase the algorithm is in), and benefits only one node per execution (i.e., only the node that initiates the execution can determine its centrality score; all other nodes cannot determine theirs and are there just to help). In contrast,

our algorithms are applicable only to tree graphs but are memory non-intensive, homogeneous (i.e., same update rules for every node at every time), and benefit all nodes in one execution (i.e., all nodes can gradually learn their scores). We also note that there are some related publications [117–119], whose aim is *not* to decentralizedly compute betweenness and closeness centralities. Rather, their aim is to introduce new centrality measures that, by definition, can be trivially decentralizedly computed and show that, in some cases, such measures correlate well with betweenness, closeness, or other meaningful measures.

The outline of this chapter is as follows: Section 3.2 derives the distributed algorithms for computing classic closeness. Section 3.3 does the same for exponential closeness. Section 3.4 simulates the behavior of the algorithms developed. Finally, Section 3.5 concludes the chapter.

3.2 Classic Closeness

To derive distributed algorithms for computing classic closeness, we first introduce some preliminaries.

3.2.1 Preliminaries

Reconsider the undirected, unweighted, and connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in Section 3.1 and suppose $\mathcal{G}$ is a tree, i.e., it has N nodes in $\mathcal{V}$ and $N-1$ edges in $\mathcal{E}$ with no cycles. Although $\mathcal{G}$ is undirected, for the purpose of development let us associate with each edge $\{i, j\} \in \mathcal{E}$ a pair of directed edges denoted as (i, j) and (j, i) (i.e., braces are for undirected edges, while parentheses are for directed ones), and let $\tilde{\mathcal{E}}$ denote the set of $2(N-1)$ directed edges.

Observe that since $\mathcal{G}$ is a tree, for each edge $\{i, j\} \in \mathcal{E}$, deleting this edge results in two connected components, one containing node j and the other containing node

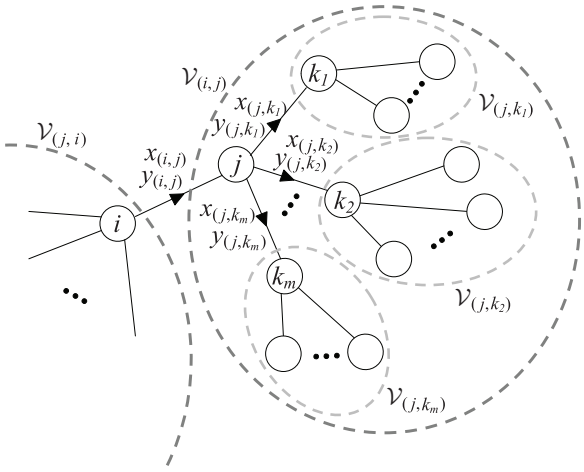


Figure 3.3: A graphical illustration of the sets $\mathcal{V}_{(i,j)}$'s, variables $x_{(i,j)}$'s, and variables $y_{(i,j)}$'s.

i. For convenience, let $\mathcal{V}_{(i,j)}$ and $\mathcal{V}_{(j,i)}$ denote, respectively, the sets of nodes in these two connected components as illustrated in (2.2)–(2.4) and in Figure 3.3, i.e.,

$$\mathcal{V}_{(i,j)} = \{k \in \mathcal{V} : k \text{ is in the connected component containing } j \text{ if } \{i,j\} \in \mathcal{E} \text{ is deleted}\}.$$

Also, $\mathcal{V}$ can be partitioned into $\mathcal{V}_{(i,j)}$ and $\mathcal{V}_{(j,i)}$, i.e.,

$$\mathcal{V}_{(i,j)} \cup \mathcal{V}_{(j,i)} = \mathcal{V} \quad \text{and} \quad \mathcal{V}_{(i,j)} \cap \mathcal{V}_{(j,i)} = \emptyset.$$

Moreover, for each node $i \in \mathcal{V}$,

$$\mathcal{V} \text{ can be partitioned into } \{i\} \text{ and } \mathcal{V}_{(i,j)} \ \forall j \in \mathcal{N}_i,$$

where $\mathcal{N}_i = \{j \in \mathcal{V} : \{i,j\} \in \mathcal{E}\}$ denotes the set of neighbors of node i.

3.2.2 Key Algebraic Relationships

With the preliminaries in hand, we next establish an algebraic relationship that expresses the classic closeness C_i's in (1.2) as functions of some variables which have graph-theoretic meaning. To achieve this goal, let us associate with each directed edge $(i,j) \in \tilde{\mathcal{E}}$ a variable $x_{(i,j)}$ representing the number of nodes in $\mathcal{V}_{(i,j)}$ as shown in (2.5) and another variable $y_{(i,j)}$ representing the sum of the distances between node i and all nodes in $\mathcal{V}_{(i,j)}$ (see Figure 3.3), i.e.,

$$\begin{aligned} x_{(i,j)} &= |\mathcal{V}_{(i,j)}|, \\ y_{(i,j)} &= \sum_{k \in \mathcal{V}_{(i,j)}} d_{ik}, \end{aligned} \tag{3.2}$$

where $\mathcal{V}_{(i,j)}$ is as defined in (2.2), $|\cdot|$ denotes the cardinality of a set, and d_{ij} again denotes the distance between nodes i and j. Then, rewrite (2.11), we have

$$\sum_{j \in \mathcal{N}_i} x_{(i,j)} = N - 1, \quad \forall i \in \mathcal{V}. \tag{3.3}$$

Moreover, due to (2.4), (3.2), and the fact that $d_{ii} = 0$, we have

$$\sum_{j \in \mathcal{V}} d_{ij} = \sum_{j \in \mathcal{N}_i} \sum_{k \in \mathcal{V}_{(i,j)}} d_{ik} = \sum_{j \in \mathcal{N}_i} y_{(i,j)}, \quad \forall i \in \mathcal{V}. \tag{3.4}$$

It follows from (3.3) and (3.4) that the classic closeness C_i of each node $i \in \mathcal{V}$ in (1.2) may be expressed as

$$C_i = \frac{\sum_{j \in \mathcal{N}_i} x_{(i,j)}}{\sum_{j \in \mathcal{N}_i} y_{(i,j)}}, \tag{3.5}$$

achieving the above goal. Note from (3.2) that $y_{(i,j)} \geq 1 \ \forall (i,j) \in \tilde{\mathcal{E}}$, so that C_i in (3.5) is always well-defined.

Expression (3.5) suggests that if each node $i \in \mathcal{V}$ is able to determine the values of the $x_{(i,j)}$'s and $y_{(i,j)}$'s associated with directed edges pointing away from it (i.e., $x_{(i,j)}$ and $y_{(i,j)}$ $\forall j \in \mathcal{N}_i$), it would be able to calculate C_i by itself. Since the $4(N-1)$ variables $x_{(i,j)}$ and $y_{(i,j)}$ $\forall (i,j) \in \tilde{\mathcal{E}}$ have graph-theoretic meaning, those associated with nearby edges may be explicitly related in some ways. Thus, if enough number of algebraic equations relating these variables can be derived, and if such equations can be decentralizedly solved by the nodes, the nodes would be able to calculate their C_i's.

Motivated by this idea, we now derive algebraic equations relating the variables. With equation (2.10)

$$x_{(i,j)} + x_{(j,i)} = N, \quad \forall (i,j) \in \tilde{\mathcal{E}},$$

and equation (2.12)

$$x_{(i,j)} - \sum_{\substack{k \in \mathcal{N}_j \\ k \neq i}} x_{(j,k)} = 1, \quad \forall (i,j) \in \tilde{\mathcal{E}}, \tag{3.6}$$

in mind, we notice that for each directed edge $(i,j) \in \tilde{\mathcal{E}}$ and each node $k \in \mathcal{V}_{(i,j)}$, the shortest path between nodes i and k must go through node j, so that $d_{ik} = d_{ij} + d_{jk}$, where $d_{ij} = 1$. Hence, $y_{(i,j)}$ in (3.2) may be written using (2.5) as

$$\begin{aligned} y_{(i,j)} &= \sum_{k \in \mathcal{V}_{(i,j)}} (1 + d_{jk}) \\ &= x_{(i,j)} + \sum_{k \in \mathcal{V}_{(i,j)}} d_{jk}. \end{aligned} \tag{3.7}$$

To cast (3.7) in a useful form, note that for each directed edge $(i,j) \in \tilde{\mathcal{E}}$, $\mathcal{V}_{(i,j)}$ can be

partitioned into $\{j\}$ and $\mathcal{V}_{(j,k)}$ $\forall k \in \mathcal{N}_j$, $k \neq i$ (see Figure 3.3). Therefore, the second term on the right-hand side of (3.7) may be stated using (3.2) as

$$\begin{aligned}\sum_{k\in\mathcal{V}_{(i,j)}} d_{jk} &= d_{jj} + \sum_{\substack{k\in\mathcal{N}_j\\k\neq i}} \sum_{\ell\in\mathcal{V}_{(j,k)}} d_{j\ell} \\ &= \sum_{\substack{k\in\mathcal{N}_j\\k\neq i}} y_{(j,k)}. \end{aligned} \tag{3.8}$$

Substituting (3.8) back into (3.7) gives

$$y_{(i,j)} - \sum_{\substack{k\in\mathcal{N}_j\\k\neq i}} y_{(j,k)} - x_{(i,j)} = 0, \quad \forall (i,j) \in \tilde{\mathcal{E}}. \tag{3.9}$$

Since $|\tilde{\mathcal{E}}| = 2(N-1)$ and since both (2.12) and (3.9) hold for every $(i,j) \in \tilde{\mathcal{E}}$, they collectively provide $4(N-1)$ *linear* equations relating the $4(N-1)$ variables $x_{(i,j)}$ and $y_{(i,j)}$ $\forall (i,j) \in \tilde{\mathcal{E}}$. Therefore, by introducing a vector $\mathbf{x} \in \mathbb{R}^{2(N-1)}$ obtained by stacking the $2(N-1)$ variables $x_{(i,j)}$'s according to some order, (2.12) can be written in a matrix form as

$$H\mathbf{x} = \mathbb{1}, \tag{3.10}$$

where $H \in \mathbb{R}^{2(N-1)\times 2(N-1)}$ is a square matrix and $\mathbb{1} \in \mathbb{R}^{2(N-1)}$ is the all-one column vector. Likewise, by introducing another vector $\mathbf{y} \in \mathbb{R}^{2(N-1)}$ obtained by stacking the $2(N-1)$ variables $y_{(i,j)}$'s in the same order as the $x_{(i,j)}$'s in $\mathbf{x}$, (3.9) can also be put into a matrix form:

$$-\mathbf{x} + H\mathbf{y} = 0. \tag{3.11}$$

Combining (3.10) and (3.11), we get

$$\underbrace{\begin{bmatrix} H & 0 \\ -I & H \end{bmatrix}}_{\mathbf{H}} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbb{1} \\ 0 \end{bmatrix}, \tag{3.12}$$

where I is the identity matrix of appropriate size and $\mathbf{H} \in \mathbb{R}^{4(N-1)\times 4(N-1)}$ is as defined in (3.12).

In Chapter 2 on betweenness computation, the matrix H also arises and is shown in (2.14) to have the form

$$H = \left[\begin{array}{cccc|cccc} 1 & * & \dots & * & 0 & 0 & \dots & 0 \\ 0 & 1 & \ddots & \vdots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & * & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & 1 & 0 & 0 & \dots & 0 \\ \hline 0 & * & \dots & * & 1 & 0 & \dots & 0 \\ * & 0 & \ddots & \vdots & * & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & * & \vdots & \ddots & \ddots & 0 \\ * & \dots & * & 0 & * & \dots & * & 1 \end{array}\right]$$

under some ordering of the elements $x_{(i,j)}$'s in the vector $\mathbf{x}$. Since H in (2.14) has a 2-by-2 block triangular structure, and since the first and second blocks on the diagonal of H are upper and lower triangular matrices with 1 on their diagonal, respectively, we see that all the $2(N-1)$ eigenvalues of H are at 1, which also makes it nonsingular. As a result, we have the following lemma and corollary:

Lemma 3. *The matrix* $\mathbf{H}$ *in* (3.12) *is a unipotent matrix with all its* $4(N-1)$ *eigenvalues at* 1.

Proof. The proof is an immediate consequence of the eigenvalues of H being all at 1

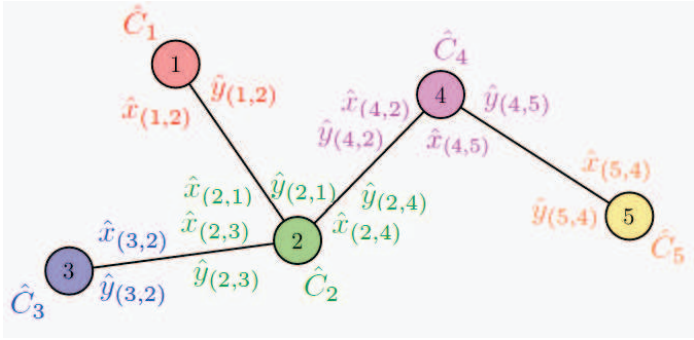


Figure 3.4: An example of how variables maintained in nodes.

and the lower block triangular structure of $\mathbf{H}$. □

Corollary 3. *The matrix $I-\mathbf{H}$ is a nilpotent matrix with all its $4(N-1)$ eigenvalues at 0.*

Proof. The proof follows immediately from the lower block triangular structure of $\mathbf{H}$ and (2.14). □

3.2.3 Continuous- and Discrete-Time Distributed Algorithms

Recall from Section 3.2.2 that if each node $i \in \mathcal{V}$ can determine the values of $x_{(i,j)}$ and $y_{(i,j)}$ $\forall j \in \mathcal{N}_i$, it could calculate C_i by itself using (3.5). In light of this fact, for each directed edge $(i,j) \in \tilde{\mathcal{E}}$, let $\hat{x}_{(i,j)}(t) \in \mathbb{R}$ and $\hat{y}_{(i,j)}(t) \in \mathbb{R}$ represent, respectively, estimates of $x_{(i,j)}$ and $y_{(i,j)}$ at time t that are maintained in node i's memory (i.e., each node $i \in \mathcal{V}$ maintains $\hat{x}_{(i,j)}(t)$ and $\hat{y}_{(i,j)}(t)$ $\forall j \in \mathcal{N}_i$). In addition, for each $i \in \mathcal{V}$, let $\hat{C}_i(t) \in \mathbb{R}$ represent an estimate of C_i at time t that is also maintained in node i's memory. An example of how variables maintained in nodes is illustrated in Figure 3.4. With these notations, consider now the following continuous-time distributed algorithm, inspired by (2.12), (3.9), and (3.5):

$$\dot{\hat{x}}_{(i,j)}(t) = -\hat{x}_{(i,j)}(t) + \sum_{\substack{k \in \mathcal{N}_j \\ k \neq i}} \hat{x}_{(j,k)}(t) + 1, \quad \forall i \in \mathcal{V},\ \forall j \in \mathcal{N}_i, \tag{3.13a}$$

$$\dot{\hat{y}}_{(i,j)}(t) = -\hat{y}_{(i,j)}(t) + \sum_{\substack{k\in\mathcal{N}_j \\ k\neq i}} \hat{y}_{(j,k)}(t) + \hat{x}_{(i,j)}(t), \qquad \forall i \in \mathcal{V},\ \forall j \in \mathcal{N}_i, \tag{3.13b}$$

$$\hat{C}_i(t) = \frac{\sum_{j\in\mathcal{N}_i} \hat{x}_{(i,j)}(t)}{\max\Big\{1, \sum_{j\in\mathcal{N}_i} \hat{y}_{(i,j)}(t)\Big\}}, \quad \forall i \in \mathcal{V}, \tag{3.13c}$$

where $t \in [0,\infty)$ denotes continuous time and $\max\{1,\cdot\}$ in (3.13c) is intended to prevent a division by zero while ensuring algorithm correctness. Observe that algorithm (3.13) is decentralizedly implementable, requiring only that each node $i \in \mathcal{V}$ repeatedly sends its $\hat{x}_{(i,j)}(t)$ and $\hat{y}_{(i,j)}(t)$ to every neighboring node $j \in \mathcal{N}_i$. Indeed, this algorithm may be viewed as a networked dynamical system with an affine state equation (3.13a) and (3.13b) and a nonlinear output equation (3.13c). Moreover, just like how (2.12) and (3.9) may be expressed as (3.12), the state equation (3.13a) and (3.13b) may be expressed in a matrix form as

$$\begin{bmatrix} \dot{\hat{\mathbf{x}}}(t) \\ \dot{\hat{\mathbf{y}}}(t) \end{bmatrix} = \underbrace{\begin{bmatrix} -H & 0 \\ I & -H \end{bmatrix}}_{-\mathbf{H}} \begin{bmatrix} \hat{\mathbf{x}}(t) \\ \hat{\mathbf{y}}(t) \end{bmatrix} + \begin{bmatrix} \mathbb{1} \\ 0 \end{bmatrix}, \tag{3.14}$$

where $\mathbf{H}$ is as in (3.12) and, like $\mathbf{x}$ and $\mathbf{y}$ earlier, $\hat{\mathbf{x}}(t) \in \mathbb{R}^{2(N-1)}$ and $\hat{\mathbf{y}}(t) \in \mathbb{R}^{2(N-1)}$ are vectors obtained by stacking the estimates $\hat{x}_{(i,j)}(t)$'s and $\hat{y}_{(i,j)}(t)$'s. The following theorem shows that the algorithm solves the classic closeness computation problem:

Theorem 5. *The continuous-time algorithm* (3.13) *has a unique equilibrium point* $(\mathbf{x},\mathbf{y})$ *that is exponentially stable, such that for any* $\hat{x}_{(i,j)}(0)$ *and* $\hat{y}_{(i,j)}(0)$ $\forall (i,j) \in \tilde{\mathcal{E}}$, *we have* $\lim_{t\to\infty} \hat{x}_{(i,j)}(t) = x_{(i,j)}$ *and* $\lim_{t\to\infty} \hat{y}_{(i,j)}(t) = y_{(i,j)}$ $\forall (i,j) \in \tilde{\mathcal{E}}$. *In addition,* $\lim_{t\to\infty} \hat{C}_i(t) = C_i$ $\forall i \in \mathcal{V}$.

Proof. Since $-\mathbf{H}$ is nonsingular due to Lemma 3, the state equation (3.14) has a unique equilibrium point. In addition, because of (3.12), the equilibrium point is

at $(\mathbf{x}, \mathbf{y})$. Furthermore, since $-\mathbf{H}$ is also Hurwitz, $(\mathbf{x}, \mathbf{y})$ is (globally) exponentially stable, such that for any $\hat{\mathbf{x}}(0) \in \mathbb{R}^{2(N-1)}$ and $\hat{\mathbf{y}}(0) \in \mathbb{R}^{2(N-1)}$, $\lim_{t\to\infty} \hat{\mathbf{x}}(t) = \mathbf{x}$ and $\lim_{t\to\infty} \hat{\mathbf{y}}(t) = \mathbf{y}$. Therefore, for each $i \in \mathcal{V}$, $\lim_{t\to\infty} \sum_{j\in\mathcal{N}_i} \hat{x}_{(i,j)}(t) = \sum_{j\in\mathcal{N}_i} x_{(i,j)}$ and $\lim_{t\to\infty} \sum_{j\in\mathcal{N}_i} \hat{y}_{(i,j)}(t) = \sum_{j\in\mathcal{N}_i} y_{(i,j)} \geq 1$, where the inequality "$\geq 1$" is a result of $y_{(i,j)} \geq 1\ \forall (i,j) \in \tilde{\mathcal{E}}$, which in turn is a result of (3.2). It follows from (3.5) and (3.13c) that $\lim_{t\to\infty} \hat{C}_i(t) = C_i\ \forall i \in \mathcal{V}$. □

Note that the continuous-time algorithm (3.13) is enabled by the unipotent property of $\mathbf{H}$ from Lemma 3. As it turns out, the nilpotent property of $I - \mathbf{H}$ from Corollary 3 is instrumental for constructing its discrete-time counterpart. Specifically, consider the following discrete-time distributed algorithm:

$$\hat{x}_{(i,j)}(t+1) = \sum_{\substack{k\in\mathcal{N}_j \\ k\neq i}} \hat{x}_{(j,k)}(t) + 1, \quad \forall i \in \mathcal{V},\ \forall j \in \mathcal{N}_i, \tag{3.15a}$$

$$\hat{y}_{(i,j)}(t+1) = \sum_{\substack{k\in\mathcal{N}_j \\ k\neq i}} \hat{y}_{(j,k)}(t) + \hat{x}_{(i,j)}(t), \quad \forall i \in \mathcal{V},\ \forall j \in \mathcal{N}_i, \tag{3.15b}$$

$$\hat{C}_i(t) = \frac{\displaystyle\sum_{j\in\mathcal{N}_i} \hat{x}_{(i,j)}(t)}{\max\Big\{1, \displaystyle\sum_{j\in\mathcal{N}_i} \hat{y}_{(i,j)}(t)\Big\}}, \quad \forall i \in \mathcal{V}, \tag{3.15c}$$

where $t \in \{0, 1, 2, \ldots\}$ here denotes discrete time. Notice that algorithm (3.15) is also a decentralizedly implementable, networked dynamical system, whose affine state equation (3.15a) and (3.15b) can be written in a matrix form as

$$\begin{bmatrix} \hat{\mathbf{x}}(t+1) \\ \hat{\mathbf{y}}(t+1) \end{bmatrix} = \underbrace{\begin{bmatrix} I - H & 0 \\ I & I - H \end{bmatrix}}_{I-\mathbf{H}} \begin{bmatrix} \hat{\mathbf{x}}(t) \\ \hat{\mathbf{y}}(t) \end{bmatrix} + \begin{bmatrix} \mathbb{1} \\ 0 \end{bmatrix}. \tag{3.16}$$

The following theorem, which makes use of the nilpotence of $I - \mathbf{H}$ from Corollary 3, shows that this algorithm not only solves the classic closeness computation problem,

it does so in finite time (i.e., it exhibits a deadbeat response):

Theorem 6. *The discrete-time algorithm* (3.15) *has a unique equilibrium point* $(\mathbf{x}, \mathbf{y})$ *that is finite-time stable, such that there exists a* $T < \infty$ *with which for any* $\hat{x}_{(i,j)}(0)$ *and* $\hat{y}_{(i,j)}(0)$ $\forall (i,j) \in \tilde{\mathcal{E}}$*, we have* $\hat{x}_{(i,j)}(t) = x_{(i,j)}$ *and* $\hat{y}_{(i,j)}(t) = y_{(i,j)}$ $\forall (i,j) \in \tilde{\mathcal{E}}$ *and* $\hat{C}_i(t) = C_i$ $\forall i \in \mathcal{V}$*, for every* $t \geq T$*.*

Proof. The proof follows the same line as the proof of Theorem 5. To show that the equilibrium point $(\mathbf{x}, \mathbf{y})$ is (globally) finite-time stable, note from (3.12) and (3.16) that

$$\begin{bmatrix} \hat{\mathbf{x}}(t+1) - \mathbf{x} \\ \hat{\mathbf{y}}(t+1) - \mathbf{y} \end{bmatrix} = \underbrace{\begin{bmatrix} I - H & 0 \\ I & I - H \end{bmatrix}}_{I - \mathbf{H}} \begin{bmatrix} \hat{\mathbf{x}}(t) - \mathbf{x} \\ \hat{\mathbf{y}}(t) - \mathbf{y} \end{bmatrix}.$$

Since $I - \mathbf{H}$ is nilpotent, there exists a $T < \infty$ such that $(I - \mathbf{H})^T = 0$, so that $\hat{\mathbf{x}}(t) = \mathbf{x}$ and $\hat{\mathbf{y}}(t) = \mathbf{y}$ $\forall t \geq T$. Hence, from (3.5), (3.15c), and the fact that $y_{(i,j)} \geq 1$ $\forall (i,j) \in \tilde{\mathcal{E}}$, we get $\hat{C}_i(t) = C_i$ $\forall i \in \mathcal{V}$ $\forall t \geq T$. □

3.3 Exponential Closeness

In this section, we shift our focus from classic closeness to exponential closeness. We show that although their definitions as given in (1.2) and (3.1) are quite different, the same approach that produces the results for classic closeness in Section 3.2 can be used here to obtain parallel results for exponential closeness.

3.3.1 Key Algebraic Relationships

As before, we first express the exponential closeness C_i^E's in (3.1) as functions of some variables with graph-theoretic meaning. To do so, let us associate with each directed

edge $(i,j) \in \tilde{\mathcal{E}}$ a variable $w_{(i,j)}$, defined as

$$w_{(i,j)} = \sum_{k \in \mathcal{V}_{(i,j)}} \alpha^{-d_{ik}}. \tag{3.17}$$

With (3.17) and (2.4), the exponential closeness C_i^E of each node $i \in \mathcal{V}$ in (3.1) becomes

$$C_i^E = \sum_{j \in \mathcal{N}_i} \sum_{k \in \mathcal{V}_{(i,j)}} \alpha^{-d_{ik}} = \sum_{j \in \mathcal{N}_i} w_{(i,j)}, \tag{3.18}$$

which is a linear function of the $w_{(i,j)}$'s. It follows that if each node $i \in \mathcal{V}$ knows the values of the $w_{(i,j)}$'s associated with its outgoing edges (i.e., $w_{(i,j)}$ $\forall j \in \mathcal{N}_i$), it could calculate its own C_i^E.

With this in mind, we next derive algebraic equations relating the $2(N-1)$ variables $w_{(i,j)}$ $\forall (i,j) \in \tilde{\mathcal{E}}$. Note from (3.17) that for each $(i,j) \in \tilde{\mathcal{E}}$, we can write

$$\begin{aligned} w_{(i,j)} &= \alpha^{-d_{ij}} + \sum_{\substack{k \in \mathcal{V}_{(i,j)} \\ k \neq j}} \alpha^{-d_{ik}} \\ &= \frac{1}{\alpha} + \sum_{\substack{k \in \mathcal{N}_j \\ k \neq i}} \sum_{l \in \mathcal{V}_{(j,k)}} \alpha^{-d_{il}} \\ &= \frac{1}{\alpha} + \sum_{\substack{k \in \mathcal{N}_j \\ k \neq i}} \sum_{l \in \mathcal{V}_{(j,k)}} \left(\alpha^{-d_{ij}} \cdot \alpha^{-d_{jl}} \right) \\ &= \frac{1}{\alpha} + \frac{1}{\alpha} \sum_{\substack{k \in \mathcal{N}_j \\ k \neq i}} \sum_{l \in \mathcal{V}_{(j,k)}} \alpha^{-d_{jl}}, \end{aligned} \tag{3.19}$$

where we have used the fact that $d_{il} = d_{ij} + d_{jl}$ and $d_{ij} = 1$. Equation (3.19), along

with (3.17), implies that

$$w_{(i,j)} - \frac{1}{\alpha} \sum_{\substack{k \in \mathcal{N}_j \\ k \neq i}} w_{(j,k)} = \frac{1}{\alpha}, \quad \forall (i,j) \in \tilde{\mathcal{E}}, \tag{3.20}$$

which is a set of $2(N-1)$ linear equations relating the $2(N-1)$ variables $w_{(i,j)}$ $\forall (i,j) \in \tilde{\mathcal{E}}$, i.e., there are as many equations as there are variables.

Next, observe that (3.20) on the $w_{(i,j)}$'s has the same form as (2.12) on the $x_{(i,j)}$'s, except that there are constant coefficients $\frac{1}{\alpha}$ appearing in (3.20) but not in (2.12). In addition, $\frac{1}{\alpha}$ appears in every term of (3.20), except for the first term on its left-hand side. Given that (2.12) can be written in a matrix form (3.10) for some H having the form of (2.14), the same can be said about (3.20). Therefore, we obtain

$$H_\alpha^E \mathbf{w} = \frac{1}{\alpha} \mathbb{1}, \tag{3.21}$$

where $H_\alpha^E \in \mathbb{R}^{2(N-1) \times 2(N-1)}$ is a square matrix having the *same* form as H in (2.14), whose entries depend on α, and $\mathbf{w} \in \mathbb{R}^{2(N-1)}$ is a vector obtained by stacking the $2(N-1)$ variables $w_{(i,j)}$'s. Because of the similarity between H_α^E and H, we obtain the following mirroring lemma and corollary:

Lemma 4. *The matrix H_α^E is a unipotent matrix with all its $2(N-1)$ eigenvalues at 1.*

Corollary 4. *The matrix $I - H_\alpha^E$ is a nilpotent matrix with all its $2(N-1)$ eigenvalues at 0.*

3.3.2 Continuous and Discrete-Time Distributed Algorithms

Leveraging the above results, we obtain the following continuous-time distributed algorithm, which is a networked dynamical system with an affine state equation and

a linear output equation:

$$\dot{\hat{\mathbf{w}}}(t) = -H_\alpha^E \hat{\mathbf{w}}(t) + \frac{1}{\alpha}\mathbb{1}, \tag{3.22a}$$

$$\hat{C}_i^E(t) = \sum_{j \in \mathcal{N}_i} \hat{w}_{(i,j)}(t), \quad \forall i \in \mathcal{V}, \tag{3.22b}$$

where here $t \in [0, \infty)$, and $\hat{\mathbf{w}}(t) = [\hat{w}_{(i,j)}(t)] \in \mathbb{R}^{2(N-1)}$ is a vector representing an estimate of $\mathbf{w} = [w_{(i,j)}]$. The theorem below, which makes use of the unipotence of H_α^E from Lemma 4, shows that this algorithm solves the exponential closeness computation problem:

Theorem 7. *The continuous-time algorithm* (3.22) *has a unique equilibrium point* $\mathbf{w}$ *that is exponentially stable, such that for any* $\hat{w}_{(i,j)}(0) \forall (i,j) \in \tilde{\mathcal{E}}$*, we have* $\lim_{t\to\infty} \hat{w}_{(i,j)}(t) = w_{(i,j)}$ $\forall (i,j) \in \tilde{\mathcal{E}}$*. In addition,* $\lim_{t\to\infty} \hat{C}_i^E(t) = C_i^E$ $\forall i \in \mathcal{V}$*.*

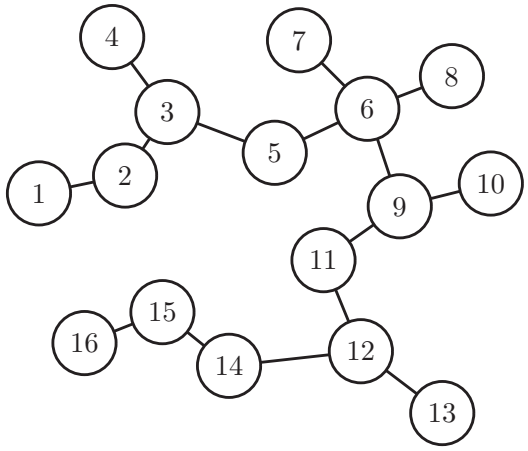
In a similar fashion, we can design a discrete-time distributed algorithm for computing exponential closeness:

$$\hat{\mathbf{w}}(t+1) = (I - H_\alpha^E)\hat{\mathbf{w}}(t) + \frac{1}{\alpha}\mathbb{1}, \tag{3.23a}$$

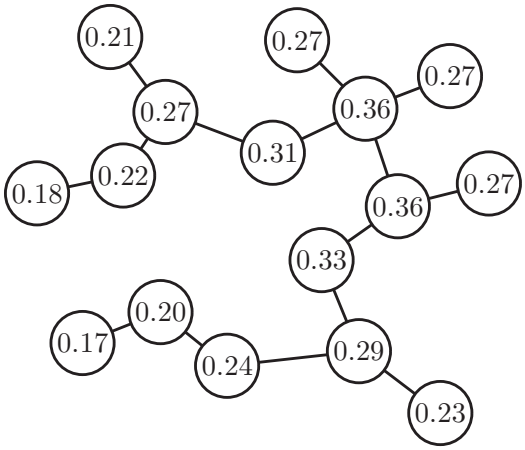
$$\hat{C}_i^E(t) = \sum_{j \in \mathcal{N}_i} \hat{w}_{(i,j)}(t), \quad \forall i \in \mathcal{V}, \tag{3.23b}$$

where here $t \in \{0, 1, 2, \ldots\}$. Finally, because $I - H_\alpha^E$ is nilpotent due to Corollary 4, we have the result below which mirrors Theorem 6:

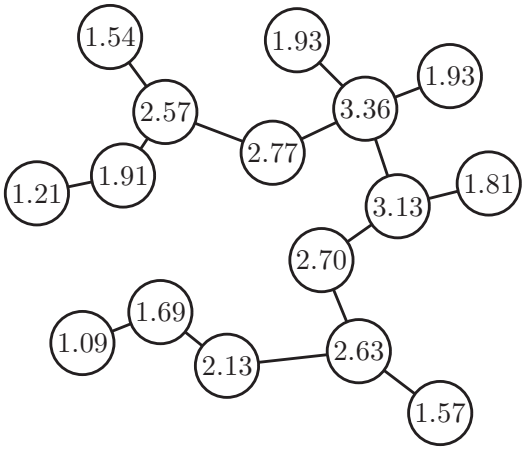
Theorem 8. *The discrete-time algorithm* (3.23) *has a unique equilibrium point* $\mathbf{w}$ *that is finite-time stable, such that there exists a* $T < \infty$ *with which for any* $\hat{w}_{(i,j)}(0)$ $\forall (i,j) \in \tilde{\mathcal{E}}$*, we have* $\hat{w}_{(i,j)}(t) = w_{(i,j)}$ $\forall (i,j) \in \tilde{\mathcal{E}}$ *and* $\hat{C}_i^E(t) = C_i^E$ $\forall i \in \mathcal{V}$*, for every* $t \geq T$*.*



(a) Node indices.



(b) Classic closeness.



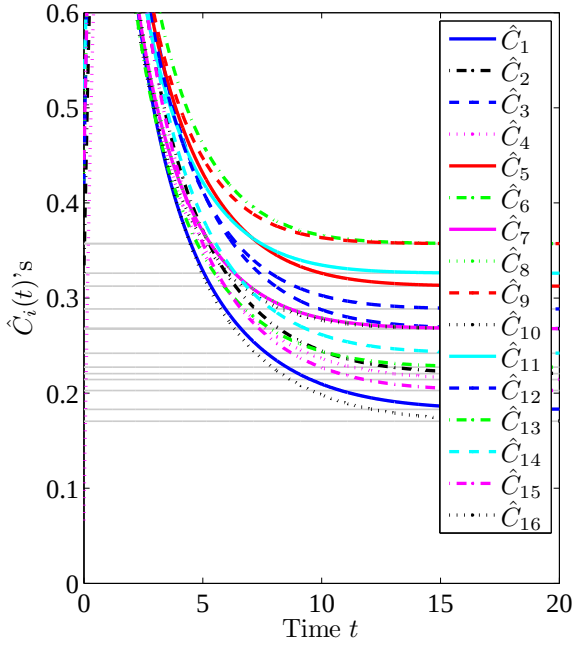
(c) Exponential closeness.

Figure 3.5: A 16-node tree graph and its node indices and classic and exponential closeness.

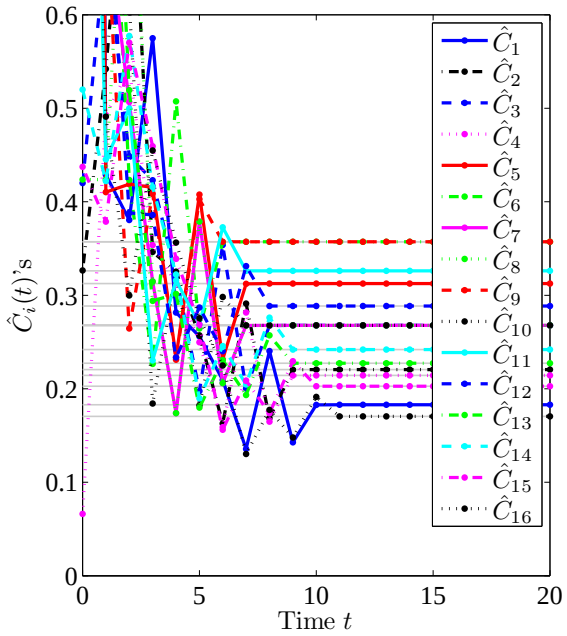
3.4 Simulation Results

In this section, we simulate the behavior of the aforementioned algorithms. To perform the simulation, consider a tree graph with $N = 16$ nodes, whose topology is shown in Figure 3.5. Also displayed in the figure are its node indices, classic closeness C_i's, and exponential closeness C_i^E's, the latter two calculated from (1.2) and (3.1) and rounded to two decimal places.

For this graph, suppose the nodes utilize the continuous-time algorithm (3.13) to simultaneously compute their estimates $\hat{C}_i(t)$'s of the unknown C_i's. In a separate



(a) Continuous-time.



(b) Discrete-time.

Figure 3.6: Performance of the continuous- and discrete-time algorithms (3.13) and (3.15) in computing classic closeness on the 16-node tree graph.

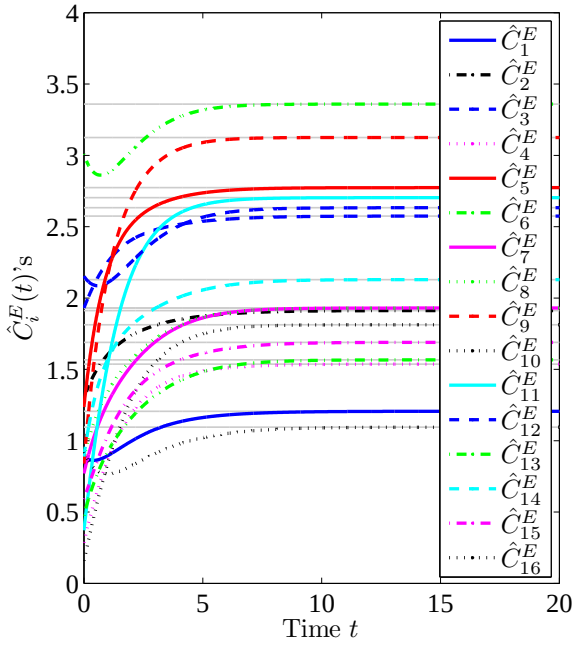
simulation, suppose they utilize the discrete-time counterpart (3.15) to do so. The simulation result is depicted in Figure 3.6, from which we see that algorithm (3.13) drives the $\hat{C}_i(t)$'s asymptotically toward the C_i's, while algorithm (3.15) forces the $\hat{C}_i(t)$'s to reach the C_i's in finite time, both agreeing with expectation. Notice that neither (3.13) nor (3.15) requires any node to build a list of distances between itself and all other nodes. Having every node build such a list would have been required, and costly, if the definition of C_i in (1.2) is used as is.

Next, suppose the nodes adopt the continuous-time algorithm (3.22) and its discrete-time counterpart (3.23) to compute their estimates $\hat{C}_i^E(t)$'s of the unknown C_i^E's. Figure 3.7 presents the simulation result, showing that the $\hat{C}_i^E(t)$'s converge to the C_i^E's asymptotically with algorithm (3.22) and in finite time with algorithm (3.23), which again is expected.

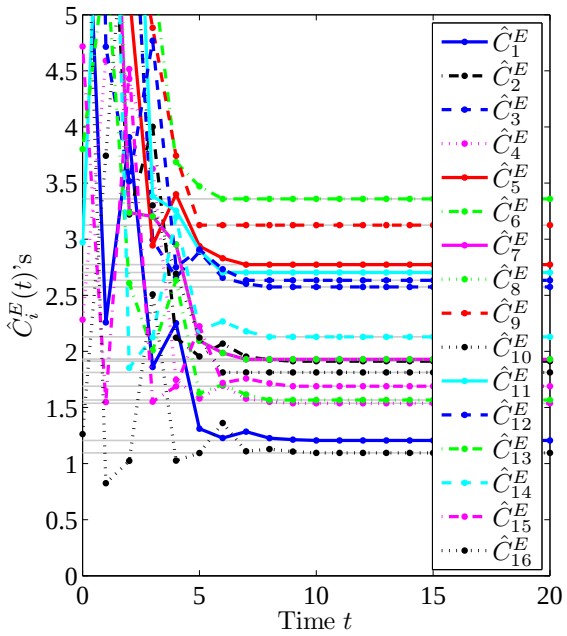
3.5 Conclusion

In this chapter, we have designed and analyzed continuous- and discrete-time distributed algorithms that allow nodes in a tree graph to cooperatively determine their classic and exponential closeness. To come up with these algorithms, we have adopted an algebraic-relationships-turned-dynamical-systems approach that was also used in Chapter 2.

Given that trees are a very special type of graphs, Chapter 5 will develop distributed algorithms for estimating closeness on general graphs, building on the ideas of this chapter.



(a) Continuous-time.



(b) Discrete-time.

Figure 3.7: Performance of the continuous- and discrete-time algorithms (3.22) and (3.23) in computing exponential closeness on the 16-node tree graph.

Chapter 4

Distributed Estimation of Betweenness Centrality

4.1 Introduction

Betweenness centrality is one of the most fundamental centrality measures [3, 6, 7] that quantifies how important a node or an edge is, within a network, based on how often it lies on the shortest paths between all pairs of nodes. Specifically, given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, \ldots, N\}$ denotes the set of N nodes and $\mathcal{E}$ the set of L edges, the *node betweenness* B_i of a node $i \in \mathcal{V}$ is defined in (1.1) [6, 7] as

$$B_i \triangleq \sum_{\substack{r \in \mathcal{V} \\ r \neq i}} \sum_{\substack{s \in \mathcal{V} \\ s \neq i, r}} \frac{\sigma(r, s, i)}{\sigma(r, s)},$$

where $\sigma(r, s)$ is the number of shortest paths from nodes r to s, and $\sigma(r, s, i)$ is the number of those that go through node i. Similarly, the *edge betweenness* $B_{\{i,j\}}$ of an

edge $\{i,j\} \in \mathcal{E}$ is defined in (2.1) [7] as

$$B_{\{i,j\}} \triangleq \sum_{r \in \mathcal{V}} \sum_{\substack{s \in \mathcal{V} \\ s \neq r}} \frac{\sigma(r,s,\{i,j\})}{\sigma(r,s)},$$

where $\sigma(r,s,\{i,j\})$ is the number of shortest paths from nodes r to s that go through edge $\{i,j\}$. Observe from (1.1) and (2.1) that the larger the number of shortest paths node i or edge $\{i,j\}$ lies on, the higher its B_i or $B_{\{i,j\}}$. Thus, B_i and $B_{\{i,j\}}$ attempt to measure how important or strategically located node i and edge $\{i,j\}$ are within the graph $\mathcal{G}$, making them useful in the analysis of complex networks [3].

Although node and edge betweenness are useful centrality measures, their computation is non-trivial because they are defined in terms of the shortest paths between all pairs of nodes. While there are classic algorithms for computing shortest paths (e.g., Floyd-Warshall algorithm [18] and Johnson's algorithm [19]), and a few for computing betweenness (e.g., [20–22]), these algorithms are *centralized* in nature, requiring that all the information about the graph $\mathcal{G}$ be available at one place, at one time, in order to execute. This requirement, unfortunately, is often difficult to meet, especially in a large network, for a variety of reasons (e.g., security, privacy, and storage). Hence, a natural question to ask is whether it is possible to compute betweenness in a *distributed* manner.

In Chapter 2, we show that when the graph $\mathcal{G}$ is a *tree*, it is possible to construct continuous- and discrete-time distributed algorithms, which enable every node $i \in \mathcal{V}$ to compute its own betweenness B_i and the betweenness $B_{\{i,j\}}$ of every edge $\{i,j\} \in \mathcal{E}$ incident on it. We also show that the algorithms are simple and scalable, require minimal node memories to execute, do not require the construction of any shortest path, and have attractive convergence properties.

In this chapter, we consider the general case where the graph $\mathcal{G}$ need not be a tree. This general case appears to be difficult to handle because there may be multiple

partially overlapping shortest paths from any node to any other node, making their counting in a distributed manner challenging. To alleviate this difficulty, in the present chapter we consider a variant of the original definition of node and edge betweenness given in (1.1) and (2.1). That is, we let the *node betweenness* B'_i of a node $i \in \mathcal{V}$ and *edge betweenness* $B'_{\{i,j\}}$ of an edge $\{i,j\} \in \mathcal{E}$ be defined as

$$B'_i \triangleq \sum_{\substack{r\in\mathcal{V}\\ r\neq i}} \sum_{\substack{s\in\mathcal{V}\\ s\neq i,r}} \mathbf{1}\Big(\frac{\sigma(r,s,i)}{\sigma(r,s)}\Big), \tag{4.1}$$

$$B'_{\{i,j\}} \triangleq \sum_{r\in\mathcal{V}} \sum_{\substack{s\in\mathcal{V}\\ s\neq r}} \mathbf{1}\Big(\frac{\sigma(r,s,\{i,j\})}{\sigma(r,s)}\Big), \tag{4.2}$$

where $\mathbf{1} : [0,\infty) \to \{0,1\}$ is an indicator function defined as $\mathbf{1}(0) = 0$ and $\mathbf{1}(z) = 1$ $\forall z > 0$. Observe that unlike B_i in (1.1) or $B_{\{i,j\}}$ in (2.1) which looks at the fraction of shortest paths from nodes r to s that go through node i or edge $\{i,j\}$, B'_i in (4.1) or $B'_{\{i,j\}}$ in (4.2) looks at whether there exists a shortest path from nodes r to s that goes through node i or edge $\{i,j\}$. Therefore, B'_i and $B'_{\{i,j\}}$ also measure how strategically located node i and edge $\{i,j\}$ are, albeit in a different way from the original B_i and $B_{\{i,j\}}$.

In this chapter, we develop a scalable continuous-time distributed algorithm, which enables every node $i \in \mathcal{V}$ to estimate its own betweenness B'_i and the betweenness $B'_{\{i,j\}}$ of every edge $\{i,j\} \in \mathcal{E}$ incident on it. To develop this algorithm, we first introduce a set of variables with graph-theoretic meaning. We then use various local properties of shortest paths to derive a set of distributed constraints on these variables. We next use these constraints to formulate a strongly convex, distributed optimization problem. We subsequently use a gradient approach to solve this problem, leading to the proposed algorithm. Finally, we evaluate the performance of the algorithm via simulation on a number of random geometric graphs, showing that it yields betweenness estimates that are fairly accurate in terms of ordering.

The outline of this chapter is as follows: Section 4.2 formulates the problem addressed. Section 4.3 derives the set of distributed constraints on betweenness. Section 4.4 derives the distributed algorithm for estimating betweenness. Section 4.5 evaluates its performance via simulation. Finally, Section 4.6 concludes the chapter.

4.2 Problem Formulation

Consider a network modeled as an undirected, unweighted and connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, \ldots, N\}$ denotes the set of $N \geq 2$ nodes and $\mathcal{E} \subset \{\{i, j\} : i, j \in \mathcal{V}, i \neq j\}$ denotes the set of L edges. Any two nodes $i, j \in \mathcal{V}$ are neighbors and can communicate if and only if $\{i, j\} \in \mathcal{E}$. The set of neighbors of each node $i \in \mathcal{V}$ is denoted as $\mathcal{N}_i = \{j \in \mathcal{V} : \{i, j\} \in \mathcal{E}\}$, and the communications are assumed to be delay- and error-free, with no quantization.

Suppose each node $i \in \mathcal{V}$ knows only the value of N and its set $\mathcal{N}_i$ of neighbors. Also suppose each node $i \in \mathcal{V}$ is willing to share its knowledge of $\mathcal{N}_i$ with every neighbor $j \in \mathcal{N}_i$, but not with others due perhaps to security and privacy reasons. Yet, despite having only such local information about the graph $\mathcal{G}$, suppose every node $i \in \mathcal{V}$ wants to determine how important it is and its incident edges are, as measured by its betweenness B'_i and its incident edge betweenness $B'_{\{i,j\}}$ $\forall j \in \mathcal{N}_i$ defined in (4.1) and (4.2).

Given the above, the goal of this chapter is to develop a scalable distributed algorithm that enables every node $i \in \mathcal{V}$ to estimate B'_i and $B'_{\{i,j\}}$ $\forall j \in \mathcal{N}_i$ with a good accuracy and without using a significant amount of memory.

4.3 Distributed Constraints on Betweenness

In this section, we lay the groundwork needed to achieve the aforementioned goal. More specifically, we begin with the introduction of a set of *basic* variables with graph-

theoretic meaning, followed by the derivation of a set of constraints on these basic variables. We then continue with the introduction of a set of *aggregated* variables that add up the basic variables, followed by the derivation of a set of constraints on these aggregated variables, which will be used in the next section to develop a distributed algorithm.

Reconsider the undirected, unweighted, and connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Although $\mathcal{G}$ is undirected, for the purpose of development let us associate with each edge $\{i, j\} \in \mathcal{E}$ a fictitious pair of directed edges denoted as (i, j) and (j, i) (i.e., braces are for undirected edges, while parentheses are for directed ones). In addition, let $\tilde{\mathcal{E}} = \{(i, j) : i, j \in \mathcal{V}, \{i, j\} \in \mathcal{E}\}$ denote the set of $2L$ directed edges. Moreover, consider the following set of *basic* variables with graph-theoretic meaning: for each $s \in \mathcal{V}$ and $(i, j) \in \tilde{\mathcal{E}}$, let

$$x_{(i,j)}^{(s)} \triangleq \begin{cases} 1, & \text{if there exists a shortest path from nodes} \\ & i \text{ to } s \text{ that goes through edge } (i, j), \\ 0, & \text{otherwise.} \end{cases} \tag{4.3}$$

For each $s \in \mathcal{V}$ and $i \in \mathcal{V}$, let $d_i^{(s)}$ denote the distance between nodes i and s. For each $r \in \mathcal{V}$, $s \in \mathcal{V}$, and $(i, j) \in \tilde{\mathcal{E}}$, let

$$b_{(i,j)}^{(r,s)} \triangleq \begin{cases} 1, & \text{if there exists a shortest path from nodes} \\ & r \text{ to } s \text{ that goes through edge } (i, j), \\ 0, & \text{otherwise.} \end{cases} \tag{4.4}$$

Finally, for each $r \in \mathcal{V}$, $s \in \mathcal{V}$, and $i \in \mathcal{V}$, let

$$b_i^{(r,s)} \triangleq \begin{cases} 1, & \text{if } r \neq i,\ s \neq i, \text{ and there exists a shortest path} \\ & \text{from nodes } r \text{ to } s \text{ that goes through node } i, \\ 0, & \text{otherwise.} \end{cases} \tag{4.5}$$

Notice from (4.5) that $b_i^{(r,s)} = 1$ implies not only that there is a shortest path from nodes r to s that goes through node i, but also that the shortest path neither begins nor ends at node i. Also note that with these basic variables, the node betweenness B'_i of a node $i \in \mathcal{V}$ in (4.1) can be rewritten as

$$B'_i = \sum_{r \in \mathcal{V}} \sum_{s \in \mathcal{V}} b_i^{(r,s)}. \tag{4.6}$$

Likewise, the edge betweenness $B'_{\{i,j\}}$ of an edge $\{i,j\} \in \mathcal{E}$ in (4.2) can be rewritten as

$$B'_{\{i,j\}} = \sum_{r \in \mathcal{V}} \sum_{s \in \mathcal{V}} (b_{(i,j)}^{(r,s)} + b_{(j,i)}^{(r,s)}). \tag{4.7}$$

Observe that for each $s \in \mathcal{V}$, (4.3) defines $2L$ binary variables $x_{(i,j)}^{(s)}$ $\forall (i,j) \in \tilde{\mathcal{E}}$. In [120], we show that these variables, together with the N distances $d_i^{(s)}$ $\forall i \in \mathcal{V}$, form a set of $2L + N$ variables that satisfy the following constraints:

Lemma 5. *For each $s \in \mathcal{V}$, the $2L + N$ variables $x_{(i,j)}^{(s)}$ $\forall (i,j) \in \tilde{\mathcal{E}}$ and $d_i^{(s)}$ $\forall i \in \mathcal{V}$ satisfy*

$$\sum_{j \in \mathcal{N}_s} x_{(s,j)}^{(s)} = 0, \tag{4.8}$$

$$\sum_{j \in \mathcal{N}_i} x_{(i,j)}^{(s)} \geq 1, \quad \forall i \in \mathcal{V}/\{s\}, \tag{4.9}$$

$$x_{(i,j)}^{(s)} + x_{(j,i)}^{(s)} \leq 1, \quad \forall \{i,j\} \in \mathcal{E}, \tag{4.10}$$

$$d_i^{(s)} - d_j^{(s)} = x_{(i,j)}^{(s)} - x_{(j,i)}^{(s)}, \quad \forall \{i,j\} \in \mathcal{E}, \tag{4.11}$$

$$d_s^{(s)} = 0. \tag{4.12}$$

Proof. Due to (4.3) and the fact that a shortest path that begins and ends at node s does not have to go through any edge, equation (4.8) holds. Since $\mathcal{G}$ is connected, for each $i \in \mathcal{V}$ with $i \neq s$, there must exist at least one shortest path from node i to node s, which must go through edge (i,j) for some $j \in \mathcal{N}_i$. Thus, $\sum_{j \in \mathcal{N}_i} x_{(i,j)}^{(s)} \geq 1$, proving (4.9). To establish (4.10), observe that for each edge $\{i,j\} \in \mathcal{E}$, if there exists a shortest path from node i to node s that goes through edge $(i,j) \in \tilde{\mathcal{E}}$, then all shortest paths from node j to node s must not go through edge $(j,i) \in \tilde{\mathcal{E}}$. Hence, $x_{(i,j)}^{(s)} + x_{(j,i)}^{(s)} \leq 1$, proving (4.10). To establish (4.11), note from (4.3) and (4.10) that for each edge $\{i,j\} \in \mathcal{E}$, $x_{(i,j)}^{(s)}$ and $x_{(j,i)}^{(s)}$ are binary but cannot be both 1. If $x_{(i,j)}^{(s)} = x_{(j,i)}^{(s)} = 0$, then nodes i and j are at the same distance from node s, so that $d_i^{(s)} = d_j^{(s)}$. If, instead, $x_{(i,j)}^{(s)} = 1$ and $x_{(j,i)}^{(s)} = 0$, then $d_i^{(s)} = d_j^{(s)} + 1$. Likewise, if $x_{(i,j)}^{(s)} = 0$ and $x_{(j,i)}^{(s)} = 1$, then $d_i^{(s)} + 1 = d_j^{(s)}$. Therefore, (4.11) holds. Finally, it is obvious that (4.12) holds. □

Theorem 11 states that the $2L + N$ variables $x_{(i,j)}^{(s)}$ $\forall (i,j) \in \tilde{\mathcal{E}}$ and $d_i^{(s)}$ $\forall i \in \mathcal{V}$ must collectively satisfy inequality and equality constraints (4.8)–(4.12). An interesting observation that can be made about these constraints is that *all of them are distributed in nature, in the sense that every constraint involves only variables which are "nearby."* Specifically, (4.8) and (4.12) involve only $x_{(s,j)}^{(s)}$ $\forall j \in \mathcal{N}_s$ and $d_s^{(s)}$, all of which are associated with node s; (4.9) involves only $x_{(i,j)}^{(s)}$ $\forall j \in \mathcal{N}_i$, all of which are associated with node i; and (4.10) and (4.11) involve only $x_{(i,j)}^{(s)}$, $x_{(j,i)}^{(s)}$, $d_i^{(s)}$, and $d_j^{(s)}$, all of which are associated with neighboring nodes i and j. In other words, none of the constraints involves variables associated with nodes who are far away from one another.

Next, observe that for each $r \in \mathcal{V}$ and $s \in \mathcal{V}$, (4.4) and (4.5) define $2L + N$ binary variables $b_{(i,j)}^{(r,s)}$ $\forall (i,j) \in \tilde{\mathcal{E}}$ and $b_i^{(r,s)}$ $\forall i \in \mathcal{V}$. The lemma below shows that these variables, together with the $2L$ variables $x_{(i,j)}^{(s)}$ $\forall (i,j) \in \tilde{\mathcal{E}}$, form a set of $4L + N$ variables that satisfy the following constraints:

Lemma 6. *For each $r \in \mathcal{V}$ and $s \in \mathcal{V}$, the $4L + N$ variables $x_{(i,j)}^{(s)}$ $\forall (i,j) \in \tilde{\mathcal{E}}$, $b_{(i,j)}^{(r,s)}$ $\forall (i,j) \in \tilde{\mathcal{E}}$, and $b_i^{(r,s)}$ $\forall i \in \mathcal{V}$ satisfy*

$$b_{(i,j)}^{(r,s)} = b_{(j,i)}^{(s,r)}, \quad \forall \{i,j\} \in \mathcal{E}, \tag{4.13}$$

$$b_{(i,j)}^{(r,s)} + b_{(j,i)}^{(r,s)} \leq 1, \quad \forall \{i,j\} \in \mathcal{E}, \tag{4.14}$$

$$b_i^{(r,s)} = b_i^{(s,r)}, \quad \forall i \in \mathcal{V}, \tag{4.15}$$

$$b_{(i,j)}^{(r,s)} = \begin{cases} x_{(i,j)}^{(s)}, & i = r, \\ x_{(i,j)}^{(s)} b_i^{(r,s)}, & i \neq r, \end{cases} \quad \forall (i,j) \in \tilde{\mathcal{E}}, \tag{4.16}$$

$$b_i^{(j,s)} = \begin{cases} 0, & i = s, \\ x_{(j,i)}^{(s)}, & i \neq s, \end{cases} \quad \forall (i,j) \in \tilde{\mathcal{E}}. \tag{4.17}$$

Proof. Let $r \in \mathcal{V}$ and $s \in \mathcal{V}$ be given. To prove (4.13), let $\{i,j\} \in \mathcal{E}$ be given. Note from (4.4) that if $b_{(i,j)}^{(r,s)} = 1$, then $b_{(j,i)}^{(s,r)} = 1$, since $\mathcal{G}$ is undirected. Conversely, if $b_{(i,j)}^{(r,s)} = 0$, then $b_{(j,i)}^{(s,r)} = 0$. Thus, (4.13) holds.

To derive (4.14), let $\{i,j\} \in \mathcal{E}$ be given. Observe from (4.4) that if $b_{(i,j)}^{(r,s)} = 1$, then all the shortest paths from nodes r to s must not go through edge (j,i). Hence, $b_{(j,i)}^{(r,s)} = 0$, implying that (4.14) holds.

To establish (4.15), let $i \in \mathcal{V}$ be given. Note from (4.5) that if $b_i^{(r,s)} = 1$, then $b_i^{(s,r)} = 1$, since $\mathcal{G}$ is undirected. Conversely, if $b_i^{(r,s)} = 0$, then $b_i^{(s,r)} = 0$. Consequently, (4.15) holds.

To prove (4.16), let $(i,j) \in \tilde{\mathcal{E}}$ be given. Notice from (4.4) and (4.3) that if $i = r$, the definitions of $b_{(i,j)}^{(r,s)}$ and $x_{(i,j)}^{(s)}$ coincide, so that $b_{(i,j)}^{(r,s)} = x_{(i,j)}^{(s)}$. Next, suppose $i \neq r$.

Note from (4.4), (4.3), and (4.5) that if $b_{(i,j)}^{(r,s)} = 1$, then $x_{(i,j)}^{(s)} = 1$ and $b_i^{(r,s)} = 1$, where the former is due to the fact that a segment of a shortest path is itself a shortest path. Conversely, if $x_{(i,j)}^{(s)} = 1$ and $b_i^{(r,s)} = 1$, then $b_{(i,j)}^{(r,s)} = 1$. Therefore, whenever $i \neq r$, we have $b_{(i,j)}^{(r,s)} = x_{(i,j)}^{(s)} b_i^{(r,s)}$. This proves (4.16).

Finally, to derive (4.17), let $(i,j) \in \tilde{\mathcal{E}}$ be given. Observe from (4.5) that if $i = s$, then $b_i^{(j,s)} = 0$. Next, suppose $i \neq s$. Note from (4.5) and (4.3) that if $b_i^{(j,s)} = 1$, then $x_{(j,i)}^{(s)} = 1$. Conversely, if $x_{(j,i)}^{(s)} = 1$, then $b_i^{(j,s)} = 1$. Hence, whenever $i \neq s$, we have $b_i^{(j,s)} = x_{(j,i)}^{(s)}$. This verifies (4.17). □

Having introduced the basic variables and derived the constraints that they must satisfy, consider next the following set of *aggregated* variables: for each $(i,j) \in \tilde{\mathcal{E}}$, let

$$x_{(i,j)} \triangleq \sum_{s \in \mathcal{V}} x_{(i,j)}^{(s)}, \tag{4.18}$$

$$B_{(i,j)} \triangleq \sum_{r \in \mathcal{V}} \sum_{s \in \mathcal{V}} b_{(i,j)}^{(r,s)}, \tag{4.19}$$

so that $x_{(i,j)}$ may be regarded as the number of nodes in $\mathcal{V}$ to which shortest paths from node i go through edge (i,j), while $B_{(i,j)}$ may be regarded as the number of ordered pairs of nodes in $\mathcal{V}$ for which there exists a shortest path from the first node to the second node that goes through edge (i,j). For each $i \in \mathcal{V}$, let

$$F_i \triangleq \sum_{s \in \mathcal{V}} d_i^{(s)}, \tag{4.20}$$

so that F_i represents the total distance between node i and all other nodes. Notice from (4.7), (4.13), and (4.19) that

$$B'_{\{i,j\}} = 2B_{(i,j)}, \quad \forall \{i,j\} \in \mathcal{E}. \tag{4.21}$$

Also note that by adding up the basic variables in the manner shown in (4.6), (4.18),

(4.19), and (4.20), we obtain a set of $4L + 2N$ aggregated variables $x_{(i,j)}\ \forall (i,j) \in \tilde{\mathcal{E}}$, $F_i\ \forall i \in \mathcal{V}$, $B_{(i,j)}\ \forall (i,j) \in \tilde{\mathcal{E}}$, as well as $B'_i\ \forall i \in \mathcal{V}$.

Next, we show that the $2L+N$ aggregated variables $x_{(i,j)}\ \forall (i,j) \in \tilde{\mathcal{E}}$ and $F_i\ \forall i \in \mathcal{V}$ satisfy the following constraints, where $|\cdot|$ denotes the cardinality of a set:

Theorem 9. *The $2L + N$ variables $x_{(i,j)}\ \forall (i,j) \in \tilde{\mathcal{E}}$ and $F_i\ \forall i \in \mathcal{V}$ satisfy*

$$\sum_{j \in \mathcal{N}_i} x_{(i,j)} \geq N - 1, \quad \forall i \in \mathcal{V}, \tag{4.22}$$

$$x_{(i,j)} + x_{(j,i)} \leq N, \quad \forall \{i,j\} \in \mathcal{E}, \tag{4.23}$$

$$F_i - F_j = x_{(i,j)} - x_{(j,i)}, \quad \forall \{i,j\} \in \mathcal{E}, \tag{4.24}$$

$$x_{(i,j)} \geq |\mathcal{N}_j / \mathcal{N}_i|, \quad \forall (i,j) \in \tilde{\mathcal{E}}, \tag{4.25}$$

$$x_{(i,j)} \leq N - |\mathcal{N}_i|, \quad \forall (i,j) \in \tilde{\mathcal{E}}, \tag{4.26}$$

$$x_{(i,j)} \leq 1 + \sum_{\substack{k \in \mathcal{N}_j / \mathcal{N}_i \\ k \neq i}} x_{(j,k)}, \quad \forall (i,j) \in \tilde{\mathcal{E}}. \tag{4.27}$$

Proof. Due to (4.9), $\sum_{j \in \mathcal{N}_i} x_{(i,j)} = \sum_{j \in \mathcal{N}_i} \sum_{s \in \mathcal{V}} x^{(s)}_{(i,j)} \geq N - 1$, proving (4.22). Based on (4.10), $x_{(i,j)} + x_{(j,i)} = \sum_{s \in \mathcal{V}} x^{(s)}_{(i,j)} + \sum_{s \in \mathcal{V}} x^{(s)}_{(j,i)} \leq N$, proving (4.23).

Equation (4.24) is an immediate result of (4.11).

For each $i \in \mathcal{V}$ and $j \in \mathcal{N}_i$, due to (4.18) and to the sets $\{j\}$ and $\mathcal{V}/\{j\}$ forming a partition of $\mathcal{V}$, we have $x_{(i,j)} = \sum_{s \in \mathcal{V}} x^{(s)}_{(i,j)} = x^{(j)}_{(i,j)} + \sum_{\substack{s \in \mathcal{V} \\ s \neq j}} x^{(s)}_{(i,j)}$. Because $x^{(j)}_{(i,j)} = 1$ according to (4.3) and because $\mathcal{N}_j / (\mathcal{N}_i \cup \{i\}) \subseteq \mathcal{V}/\{j\}$,

$$x_{(i,j)} \geq 1 + \sum_{s \in \mathcal{N}_j / (\mathcal{N}_i \cup \{i\})} x^{(s)}_{(i,j)}. \tag{4.28}$$

Moreover, because for any $s \in \mathcal{N}_j / (\mathcal{N}_i \cup \{i\})$, there exists a shortest path from node i to node s going through edge (i,j), we have $x^{(s)}_{(i,j)} = 1\ \forall s \in \mathcal{N}_j / (\mathcal{N}_i \cup \{i\})$. As a result, $\sum_{s \in \mathcal{N}_j / (\mathcal{N}_i \cup \{i\})} x^{(s)}_{(i,j)} = |\mathcal{N}_j / \mathcal{N}_i| - 1$. Substituting it into (4.28), we obtain (4.25).

For each $i \in \mathcal{V}$ and $j \in \mathcal{N}_i$, let $\mathcal{V}$ be partitioned into $\{i\}$, $\mathcal{N}_i/\{j\}$, and $\widetilde{\mathcal{V}}_{ij}$.

According to (4.18), we have $x_{(i,j)} = \sum_{s\in\mathcal{V}} x^{(s)}_{(i,j)} = x^{(i)}_{(i,j)} + \sum_{\substack{s\in\mathcal{N}_i \\ s\neq j}} x^{(s)}_{(i,j)} + \sum_{s\in\widetilde{\mathcal{V}}_{ij}} x^{(s)}_{(i,j)}$. Because $x^{(i)}_{(i,j)} = 0$ and because for any $s \in \mathcal{N}_i/\{j\}$ the shortest path from node i to node s must not go through edge (i,j), we have $\sum_{\substack{s\in\mathcal{N}_i \\ s\neq j}} x^{(s)}_{(i,j)} = 0$. Thus, $x_{(i,j)} = \sum_{s\in\widetilde{\mathcal{V}}_{ij}} x^{(s)}_{(i,j)}$. Since $|\widetilde{\mathcal{V}}_{ij}| \leq N - |\mathcal{N}_i|$ and since $x^{(s)}_{(i,j)} \leq 1\ \forall s \in \mathcal{V}$, we have $x_{(i,j)} \leq N - |\mathcal{N}_i|$, proving (4.26).

For each $i \in \mathcal{V}$ and $j \in \mathcal{N}_i$, let $\mathcal{N}_j$ be partitioned into $\mathcal{N}_j/(\mathcal{N}_i \cup \{i\})$, $\mathcal{N}_i \cap \mathcal{N}_j$, and $\{i\}$. According to (4.9), for each $s \in \mathcal{V}$ with $s \neq j$, we have $\sum_{k\in\mathcal{N}_j} x^{(s)}_{(j,k)} = \sum_{k\in\mathcal{N}_j/(\mathcal{N}_i\cup\{i\})} x^{(s)}_{(j,k)} + \sum_{k\in\mathcal{N}_i\cap\mathcal{N}_j} x^{(s)}_{(j,k)} + x^{(s)}_{(j,i)} \geq 1$. This expression can be rewritten using (4.10) as

$$x^{(s)}_{(i,j)} \leq \sum_{k\in\mathcal{N}_j/(\mathcal{N}_i\cup\{i\})} x^{(s)}_{(j,k)} + \sum_{k\in\mathcal{N}_i\cap\mathcal{N}_j} x^{(s)}_{(j,k)}. \tag{4.29}$$

Now suppose there exists a shortest path from node i to node s that goes through node j, i.e., $x^{(s)}_{(i,j)} = 1$. We claim that the shortest path must not go through edge $(j,k)\ \forall k \in \mathcal{N}_i \cap \mathcal{N}_j$. Assume to the contrary that it does, the shortest path must take the form $(i,j,k,\ldots,s)$ for some $k \in \mathcal{N}_i \cap \mathcal{N}_j$. Then, because node k is also a neighbor of node i, the path $(i,k,\ldots,s)$ is shorter than $(i,j,k,\ldots,s)$, which is a contradiction. Thus, $\forall s \in \mathcal{V}$ with $s \neq j$, when $x^{(s)}_{(i,j)} = 1$, we have $\sum_{k\in\mathcal{N}_i\cap\mathcal{N}_j} x^{(s)}_{(j,k)} = 0$. Hence, due to (4.29), we have $x^{(s)}_{(i,j)} \leq \sum_{k\in\mathcal{N}_j/(\mathcal{N}_i\cup\{i\})} x^{(s)}_{(j,k)}\ \forall s \in \mathcal{V}$ with $s \neq j$. Finally, by aggregating s over $\mathcal{V}$, we have $x_{(i,j)} = \sum_{s\in\mathcal{V}} x^{(s)}_{(i,j)} = x^{(j)}_{(i,j)} + \sum_{\substack{s\in\mathcal{V} \\ s\neq j}} x^{(s)}_{(i,j)} \leq 1 + \sum_{s\in\mathcal{V}} \sum_{k\in\mathcal{N}_j/(\mathcal{N}_i\cup\{i\})} x^{(s)}_{(j,k)} = 1 + \sum_{k\in\mathcal{N}_j/(\mathcal{N}_i\cup\{i\})} x_{(j,k)}$, proving (4.27). □

Note that constraints (4.22)–(4.27) are distributed in nature, in the sense that each constraint involves only aggregated variables that are "nearby" in the graph $\mathcal{G}$. The following theorem presents an additional set of constraints on the aggregated variables, which are also distributed:

Theorem 10. *The $4L+N$ variables $x_{(i,j)}\ \forall (i,j) \in \tilde{\mathcal{E}}$, $B_{(i,j)}\ \forall (i,j) \in \tilde{\mathcal{E}}$, and $B'_i\ \forall i \in \mathcal{V}$*

satisfy

$$B_{(i,j)} = B_{(j,i)}, \quad \forall \{i,j\} \in \mathcal{E}, \tag{4.30}$$

$$B_{(i,j)} \leq x_{(i,j)} x_{(j,i)}, \quad \forall (i,j) \in \tilde{\mathcal{E}}, \tag{4.31}$$

$$B_{(i,j)} \geq x_{(i,j)} + x_{(j,i)} - 1, \quad \forall (i,j) \in \tilde{\mathcal{E}}, \tag{4.32}$$

$$B_{(i,j)} - x_{(i,j)} \leq \sum_{\substack{h \in \mathcal{N}_i \\ h \neq j}} (B_{(i,h)} - x_{(i,h)}), \quad \forall (i,j) \in \tilde{\mathcal{E}}, \tag{4.33}$$

$$B'_i - B_{(i,j)} + x_{(i,j)} \geq \sum_{\substack{h \in \mathcal{N}_i / \mathcal{N}_j \\ h \neq j}} (x_{(h,i)} - 1), \quad \forall (i,j) \in \tilde{\mathcal{E}}, \tag{4.34}$$

$$B'_i \leq \sum_{j \in \mathcal{N}_i} \sum_{\substack{h \in \mathcal{N}_i \\ h \neq j}} x_{(i,j)} x_{(i,h)}, \quad \forall i \in \mathcal{V}, \tag{4.35}$$

$$B'_i \geq \sum_{j \in \mathcal{N}_i} x_{(j,i)} - |\mathcal{N}_i|, \quad \forall i \in \mathcal{V}, \tag{4.36}$$

$$B'_i \leq \sum_{j \in \mathcal{N}_i} (B_{(i,j)} - x_{(i,j)}), \quad \forall i \in \mathcal{V}, \tag{4.37}$$

$$2B_{(i,j)} - B'_i \leq 2x_{(i,j)}, \quad \forall (i,j) \in \tilde{\mathcal{E}}, \tag{4.38}$$

$$B'_i - 2B_{(i,j)} \leq (N - x_{(i,j)})^2 - 2x_{(i,j)}, \quad \forall (i,j) \in \tilde{\mathcal{E}}. \tag{4.39}$$

Proof. To prove (4.30), let $\{i,j\} \in \mathcal{E}$ be given. Then, according to (4.19) and (4.13),

$$\begin{aligned} B_{(i,j)} &= \sum_{r \in \mathcal{V}} \sum_{s \in \mathcal{V}} b^{(r,s)}_{(i,j)} \\ &= \sum_{s \in \mathcal{V}} \sum_{r \in \mathcal{V}} b^{(s,r)}_{(j,i)} \\ &= B_{(j,i)}, \end{aligned}$$

proving (4.30).

To derive (4.31), let $(i,j) \in \tilde{\mathcal{E}}$ be given. Notice from (4.16) that $b^{(r,s)}_{(i,j)} \leq x^{(s)}_{(i,j)}$ $\forall r \in \mathcal{V}$ $\forall s \in \mathcal{V}$. Also note from (4.13) and (4.16) that $b^{(r,s)}_{(i,j)} = b^{(s,r)}_{(j,i)} \leq x^{(r)}_{(j,i)}$ $\forall r \in \mathcal{V}$

$\forall s \in \mathcal{V}$. Thus, $b_{(i,j)}^{(r,s)} \leq x_{(i,j)}^{(s)} x_{(j,i)}^{(r)}$ $\forall r \in \mathcal{V}$ $\forall s \in \mathcal{V}$. It follows from (4.19) and (4.18) that

$$\begin{aligned} B_{(i,j)} &= \sum_{r \in \mathcal{V}} \sum_{s \in \mathcal{V}} b_{(i,j)}^{(r,s)} \\ &\leq \sum_{r \in \mathcal{V}} \sum_{s \in \mathcal{V}} x_{(i,j)}^{(s)} x_{(j,i)}^{(r)} \\ &= \sum_{s \in \mathcal{V}} x_{(i,j)}^{(s)} \sum_{r \in \mathcal{V}} x_{(j,i)}^{(r)} \\ &= x_{(i,j)} x_{(j,i)}. \end{aligned}$$

Hence, (4.31) holds.

To establish (4.32), let $(i,j) \in \tilde{\mathcal{E}}$ be given. Then, because of (4.19), (4.13), (4.16), and (4.18),

$$\begin{aligned} B_{(i,j)} &= \sum_{r \in \mathcal{V}} \sum_{s \in \mathcal{V}} b_{(i,j)}^{(r,s)} \\ &\geq \sum_{s \in \mathcal{V}} b_{(i,j)}^{(i,s)} + \sum_{r \in \mathcal{V}} b_{(i,j)}^{(r,j)} - b_{(i,j)}^{(i,j)} \\ &= \sum_{s \in \mathcal{V}} b_{(i,j)}^{(i,s)} + \sum_{r \in \mathcal{V}} b_{(j,i)}^{(j,r)} - 1 \\ &= \sum_{s \in \mathcal{V}} x_{(i,j)}^{(s)} + \sum_{r \in \mathcal{V}} x_{(j,i)}^{(r)} - 1 \\ &= x_{(i,j)} + x_{(j,i)} - 1, \end{aligned}$$

establishing (4.32).

To prove (4.33), let $(i,j) \in \tilde{\mathcal{E}}$ be given. Then, because of (4.30), (4.19), (4.18), (4.16), and (4.13),

$$\begin{aligned} B_{(i,j)} - x_{(i,j)} &= B_{(j,i)} - x_{(i,j)} \\ &= \sum_{r \in \mathcal{V}} \sum_{s \in \mathcal{V}} b_{(j,i)}^{(r,s)} - \sum_{r \in \mathcal{V}} x_{(i,j)}^{(r)} \end{aligned}$$

$$
\begin{aligned}
&= \sum_{r\in\mathcal{V}} \sum_{s\in\mathcal{V}} b_{(j,i)}^{(r,s)} - \sum_{r\in\mathcal{V}} b_{(i,j)}^{(i,r)} \\
&= \sum_{r\in\mathcal{V}} \sum_{s\in\mathcal{V}} b_{(j,i)}^{(r,s)} - \sum_{r\in\mathcal{V}} b_{(j,i)}^{(r,i)} \\
&= \sum_{r\in\mathcal{V}} \sum_{\substack{s\in\mathcal{V}\\ s\neq i}} b_{(j,i)}^{(r,s)}.
\end{aligned}
$$

Notice that for each $s \in \mathcal{V}$ with $s \neq i$, $b_{(j,i)}^{(i,s)} = 0$. Also note that for each $r \in \mathcal{V}$ and $s \in \mathcal{V}$ with $s \neq i$, if $b_{(j,i)}^{(r,s)} = 1$, then there exists $h \in \mathcal{N}_i$ with $h \neq j$ such that $b_{(i,h)}^{(r,s)} = 1$. These, along with (4.19), (4.16), and (4.18), imply that

$$
\begin{aligned}
B_{(i,j)} - x_{(i,j)} &= \sum_{\substack{r\in\mathcal{V}\\ r\neq i}} \sum_{\substack{s\in\mathcal{V}\\ s\neq i}} b_{(j,i)}^{(r,s)} \\
&\leq \sum_{\substack{r\in\mathcal{V}\\ r\neq i}} \sum_{\substack{s\in\mathcal{V}\\ s\neq i}} \sum_{\substack{h\in\mathcal{N}_i\\ h\neq j}} b_{(i,h)}^{(r,s)} \\
&= \sum_{\substack{h\in\mathcal{N}_i\\ h\neq j}} \sum_{\substack{r\in\mathcal{V}\\ r\neq i}} \sum_{\substack{s\in\mathcal{V}\\ s\neq i}} b_{(i,h)}^{(r,s)} \\
&= \sum_{\substack{h\in\mathcal{N}_i\\ h\neq j}} \sum_{\substack{r\in\mathcal{V}\\ r\neq i}} \sum_{s\in\mathcal{V}} b_{(i,h)}^{(r,s)} \\
&= \sum_{\substack{h\in\mathcal{N}_i\\ h\neq j}} \Big(\sum_{r\in\mathcal{V}} \sum_{s\in\mathcal{V}} b_{(i,h)}^{(r,s)} - \sum_{s\in\mathcal{V}} b_{(i,h)}^{(i,s)} \Big) \\
&= \sum_{\substack{h\in\mathcal{N}_i\\ h\neq j}} (B_{(i,h)} - x_{(i,h)}),
\end{aligned}
$$

proving (4.33).

To derive (4.34), let $(i,j) \in \tilde{\mathcal{E}}$ be given. Then, due to (4.19), (4.16), (4.18), and the fact that $b_i^{(i,s)} = 0$,

$$
B_{(i,j)} = \sum_{s\in\mathcal{V}} \sum_{r\in\mathcal{V}} b_{(i,j)}^{(r,s)}
$$

$$= \sum_{s\in\mathcal{V}} x^{(s)}_{(i,j)} + \sum_{s\in\mathcal{V}} \sum_{\substack{r\in\mathcal{V}\\ r\neq i}} x^{(s)}_{(i,j)} b_i^{(r,s)}$$

$$= x_{(i,j)} + \sum_{r\in\mathcal{V}} \sum_{s\in\mathcal{V}} x^{(s)}_{(i,j)} b_i^{(r,s)}. \tag{4.40}$$

In addition, because of (4.6), (4.10), and (4.40),

$$\begin{aligned} B_i' &= \sum_{r\in\mathcal{V}} \sum_{s\in\mathcal{V}} b_i^{(r,s)} \\ &\geq \sum_{r\in\mathcal{V}} \sum_{s\in\mathcal{V}} (x^{(s)}_{(i,j)} + x^{(s)}_{(j,i)}) b_i^{(r,s)} \\ &= \sum_{r\in\mathcal{V}} \sum_{s\in\mathcal{V}} x^{(s)}_{(i,j)} b_i^{(r,s)} + \sum_{r\in\mathcal{V}} \sum_{s\in\mathcal{V}} x^{(s)}_{(j,i)} b_i^{(r,s)} \\ &= B_{(i,j)} - x_{(i,j)} + \sum_{r\in\mathcal{V}} \sum_{s\in\mathcal{V}} x^{(s)}_{(j,i)} b_i^{(r,s)} \\ &\geq B_{(i,j)} - x_{(i,j)} + \sum_{r\in\mathcal{V}} \sum_{\substack{h\in\mathcal{N}_i/\mathcal{N}_j\\ h\neq j}} x^{(h)}_{(j,i)} b_i^{(r,h)}. \end{aligned} \tag{4.41}$$

Furthermore, notice that for each $h \in \mathcal{N}_i/\mathcal{N}_j$ with $h \neq j$, we have $x^{(h)}_{(j,i)} = 1$. This, along with (4.15), (4.17), and (4.18), allows us to write

$$\begin{aligned} \sum_{r\in\mathcal{V}} \sum_{\substack{h\in\mathcal{N}_i/\mathcal{N}_j\\ h\neq j}} x^{(h)}_{(j,i)} b_i^{(r,h)} &= \sum_{r\in\mathcal{V}} \sum_{\substack{h\in\mathcal{N}_i/\mathcal{N}_j\\ h\neq j}} b_i^{(r,h)} \\ &= \sum_{\substack{h\in\mathcal{N}_i/\mathcal{N}_j\\ h\neq j}} \sum_{r\in\mathcal{V}} b_i^{(h,r)} \\ &= \sum_{\substack{h\in\mathcal{N}_i/\mathcal{N}_j\\ h\neq j}} \sum_{\substack{r\in\mathcal{V}\\ r\neq i}} x^{(r)}_{(h,i)} \\ &= \sum_{\substack{h\in\mathcal{N}_i/\mathcal{N}_j\\ h\neq j}} (x_{(h,i)} - 1). \end{aligned} \tag{4.42}$$

Combining (4.41) and (4.42), we arrive at (4.34).

To establish (4.35), let $i \in \mathcal{V}$ be given. Observe from (4.5) and (4.3) that if

$b_i^{(r,s)} = 1$, then there exists $j' \in \mathcal{N}_i$ such that $x_{(i,j')}^{(s)} = 1$, and there exists $h' \in \mathcal{N}_i$ with $h' \neq j'$ such that $x_{(i,h')}^{(r)} = 1$. Hence, $b_i^{(r,s)} \leq \sum_{j \in \mathcal{N}_i} \sum_{\substack{h \in \mathcal{N}_i \\ h \neq j}} x_{(i,j)}^{(s)} x_{(i,h)}^{(r)}$. It follows from (4.6) and (4.18) that

$$\begin{aligned} B_i' &= \sum_{r \in \mathcal{V}} \sum_{s \in \mathcal{V}} b_i^{(r,s)} \\ &\leq \sum_{r \in \mathcal{V}} \sum_{s \in \mathcal{V}} \sum_{j \in \mathcal{N}_i} \sum_{\substack{h \in \mathcal{N}_i \\ h \neq j}} x_{(i,j)}^{(s)} x_{(i,h)}^{(r)} \\ &= \sum_{j \in \mathcal{N}_i} \sum_{\substack{h \in \mathcal{N}_i \\ h \neq j}} \sum_{s \in \mathcal{V}} x_{(i,j)}^{(s)} \sum_{r \in \mathcal{V}} x_{(i,h)}^{(r)} \\ &= \sum_{j \in \mathcal{N}_i} \sum_{\substack{h \in \mathcal{N}_i \\ h \neq j}} x_{(i,j)} x_{(i,h)}, \end{aligned}$$

which is exactly (4.35).

To prove (4.36), let $i \in \mathcal{V}$ be given. Then, due to (4.6), (4.17), (4.18), and the fact that $x_{(j,i)}^{(i)} = 1$,

$$\begin{aligned} B_i' &= \sum_{r \in \mathcal{V}} \sum_{s \in \mathcal{V}} b_i^{(r,s)} \\ &\geq \sum_{j \in \mathcal{N}_i} \sum_{s \in \mathcal{V}} b_i^{(j,s)} \\ &= \sum_{j \in \mathcal{N}_i} \sum_{\substack{s \in \mathcal{V} \\ s \neq i}} x_{(j,i)}^{(s)} \\ &= \sum_{j \in \mathcal{N}_i} \Big(\sum_{s \in \mathcal{V}} x_{(j,i)}^{(s)} - x_{(j,i)}^{(i)} \Big) \\ &= \sum_{j \in \mathcal{N}_i} x_{(j,i)} - |\mathcal{N}_i|, \end{aligned}$$

proving (4.36).

To derive (4.37), let $i \in \mathcal{V}$ be given. Then, due to (4.6), the fact that $b_i^{(r,i)} = 0$,

(4.9), and (4.40),

$$\begin{aligned} B_i' &= \sum_{r\in\mathcal{V}}\sum_{s\in\mathcal{V}} b_i^{(r,s)} \\ &\le \sum_{r\in\mathcal{V}}\sum_{s\in\mathcal{V}} \Big(\sum_{j\in\mathcal{N}_i} x_{(i,j)}^{(s)}\Big) b_i^{(r,s)} \\ &= \sum_{j\in\mathcal{N}_i}\sum_{r\in\mathcal{V}}\sum_{s\in\mathcal{V}} x_{(i,j)}^{(s)} b_i^{(r,s)} \\ &= \sum_{j\in\mathcal{N}_i} (B_{(i,j)} - x_{(i,j)}), \end{aligned}$$

so that (4.37) holds.

To establish (4.38), let $(i,j) \in \tilde{\mathcal{E}}$ be given. Observe from (4.16) that for each $r \in \mathcal{V}$ and $s \in \mathcal{V}$ with $r \neq i$, we have $x_{(i,j)}^{(s)} b_i^{(r,s)} = (x_{(i,j)}^{(s)} b_i^{(r,s)}) b_i^{(r,s)} = b_{(i,j)}^{(r,s)} b_i^{(r,s)}$. Moreover, $b_i^{(i,s)} = 0$. Thus, in view of (4.40),

$$B_{(i,j)} - x_{(i,j)} = \sum_{r\in\mathcal{V}}\sum_{s\in\mathcal{V}} b_{(i,j)}^{(r,s)} b_i^{(r,s)}. \tag{4.43}$$

Applying (4.13) and (4.15) to (4.43) and interchanging r and s, we get

$$\begin{aligned} B_{(i,j)} - x_{(i,j)} &= \sum_{r\in\mathcal{V}}\sum_{s\in\mathcal{V}} b_{(j,i)}^{(s,r)} b_i^{(s,r)} \\ &= \sum_{r\in\mathcal{V}}\sum_{s\in\mathcal{V}} b_{(j,i)}^{(r,s)} b_i^{(r,s)}. \end{aligned} \tag{4.44}$$

Adding up (4.43) and (4.44), and using (4.14) and (4.6), we obtain

$$\begin{aligned} 2B_{(i,j)} - 2x_{(i,j)} &= \sum_{r\in\mathcal{V}}\sum_{s\in\mathcal{V}} (b_{(i,j)}^{(r,s)} + b_{(j,i)}^{(r,s)}) b_i^{(r,s)} \\ &\le \sum_{r\in\mathcal{V}}\sum_{s\in\mathcal{V}} b_i^{(r,s)} \\ &= B_i', \end{aligned}$$

establishing (4.38).

Finally, due to space limitation, the proof of (4.39) is omitted. □

To summarize, we have introduced in this section a set of $4L + 2N$ aggregated variables $x_{(i,j)}\ \forall (i,j) \in \tilde{\mathcal{E}}$, $F_i\ \forall i \in \mathcal{V}$, $B_{(i,j)}\ \forall (i,j) \in \tilde{\mathcal{E}}$, and $B'_i\ \forall i \in \mathcal{V}$ along with a set of distributed constraints on them.

4.4 Distributed Estimation of Betweenness

In this section, we use the aggregated variables and distributed constraints to develop a distributed algorithm for estimating the unknown node and edge betweenness.

We begin by describing the idea behind the algorithm. First, observe that the $4L + 2N$ aggregated variables $x_{(i,j)}\ \forall (i,j) \in \tilde{\mathcal{E}}$, $F_i\ \forall i \in \mathcal{V}$, $B_{(i,j)}\ \forall (i,j) \in \tilde{\mathcal{E}}$, and $B'_i\ \forall i \in \mathcal{V}$ must satisfy a set of 16 distributed constraints, of which the first 6 come from Theorem 9 (i.e., (4.22)–(4.27)), and the next 10 come from Theorem 10 (i.e., (4.30)–(4.39)). Next, suppose each node $i \in \mathcal{V}$ at each time $t \in [0, \infty)$ maintains in its local memory an estimate $\hat{x}_{(i,j)}(t) \in \mathbb{R}$ of $x_{(i,j)}$ for every $j \in \mathcal{N}_i$, an estimate $\hat{F}_i(t) \in \mathbb{R}$ of F_i, an estimate $\hat{B}_{(i,j)}(t) \in \mathbb{R}$ of $B_{(i,j)}$ for every $j \in \mathcal{N}_i$, and an estimate $\hat{B}_i(t) \in \mathbb{R}$ of B'_i for a low memory complexity of $O(|\mathcal{N}_i|)$. Moreover, suppose a distributed algorithm can be designed, with which the N nodes are able to cooperatively update their estimates, driving them towards simultaneously satisfying the 16 constraints. Then, every node $i \in \mathcal{V}$ would be able to estimate B'_i and $B'_{\{i,j\}}\ \forall j \in \mathcal{N}_i$, the latter using (4.21). Indeed, the tighter the 16 constraints, the more accurate the estimates would be, which explains why we attempt to derive as many constraints as possible in Theorems 9 and 10.

To realize the above idea, consider the following notations: let $\mathbf{x} \in \mathbb{R}^{2L}$ be a vector formed by the $2L$ variables $x_{(i,j)}\ \forall (i,j) \in \tilde{\mathcal{E}}$, $\mathbf{F} \in \mathbb{R}^N$ be a vector formed by the N variables $F_i\ \forall i \in \mathcal{V}$, and $\mathbf{B} \in \mathbb{R}^{2L+N}$ be a vector formed by the $2L+N$ variables $B_{(i,j)}$

$\forall (i,j) \in \tilde{\mathcal{E}}$ and $B_i' \; \forall i \in \mathcal{V}$. Similarly, let $\hat{\mathbf{x}}(t) \in \mathbb{R}^{2L}$, $\hat{\mathbf{F}}(t) \in \mathbb{R}^N$, and $\hat{\mathbf{B}}(t) \in \mathbb{R}^{2L+N}$ be vectors formed by the $4L+2N$ estimates $\hat{x}_{(i,j)}(t) \; \forall (i,j) \in \tilde{\mathcal{E}}$, $\hat{F}_i(t) \; \forall i \in \mathcal{V}$, $\hat{B}_{(i,j)}(t)$ $\forall (i,j) \in \tilde{\mathcal{E}}$, and $\hat{B}_i(t) \; \forall i \in \mathcal{V}$. In addition, let $\Theta \subset \mathbb{R}^{2L} \times \mathbb{R}^N \times \mathbb{R}^{2L+N}$ denote the feasible set of points that satisfy the 16 constraints from Theorems 9 and 10. Clearly, the aggregated variables $(\mathbf{x}, \mathbf{F}, \mathbf{B})$ are in the set Θ.

With the notations in hand, we now address the question of how to design a distributed algorithm, which enables the N nodes to cooperatively drive their estimates $(\hat{\mathbf{x}}(t), \hat{\mathbf{F}}(t), \hat{\mathbf{B}}(t))$ into the set Θ, in which the aggregated variables $(\mathbf{x}, \mathbf{F}, \mathbf{B})$ reside. To this end, observe that the set Θ is a *non-convex* set in $\mathbb{R}^{2L} \times \mathbb{R}^N \times \mathbb{R}^{2L+N}$, which may complicate the design. The set Θ is non-convex because some of the 16 constraints—specifically, (4.31), (4.35), and (4.39)—involve products of the $x_{(i,j)}$'s. Fortunately, however, the 6 constraints from Theorem 9 involve only $(\mathbf{x}, \mathbf{F})$ and not $\mathbf{B}$. In addition, these constraints define a set Θ_1 in $\mathbb{R}^{2L} \times \mathbb{R}^N$ that is *convex* in $(\mathbf{x}, \mathbf{F})$. In comparison, the 10 constraints from Theorem 10 involve only $(\mathbf{x}, \mathbf{B})$ and not $\mathbf{F}$. Moreover, for each *fixed* $\mathbf{x}$, these constraints define a set $\Theta_2(\mathbf{x})$ in $\mathbb{R}^{2L+N}$ that is *convex* in $\mathbf{B}$.

The above observations suggest that we may construct a distributed algorithm consisting of two parts, in which the first part forces $(\hat{\mathbf{x}}(t), \hat{\mathbf{F}}(t))$ to converge to a point $(\mathbf{x}^*, \mathbf{F}^*)$ in the convex set Θ_1, while the second part forces $\hat{\mathbf{B}}(t)$ to converge to a point in the time-varying convex set $\Theta_2(\hat{\mathbf{x}}(t))$. Since $\hat{\mathbf{x}}(t)$ will converge to $\mathbf{x}^*$ as $t \to \infty$, the time-varying convex set $\Theta_2(\hat{\mathbf{x}}(t))$ will converge to the time-invariant convex set $\Theta_2(\mathbf{x}^*)$, so that $\hat{\mathbf{B}}(t)$ will converge to a point $\mathbf{B}^*(\mathbf{x}^*)$ in $\Theta_2(\mathbf{x}^*)$. It follows that $(\hat{\mathbf{x}}(t), \hat{\mathbf{F}}(t), \hat{\mathbf{B}}(t))$ will converge to the point $(\mathbf{x}^*, \mathbf{F}^*, \mathbf{B}^*(\mathbf{x}^*))$ in the set Θ, despite its non-convexity.

To devise a distributed algorithm that has the aforementioned features, consider the following two optimization problems:

$$\min_{(\hat{\mathbf{x}}, \hat{\mathbf{F}}) \in \mathbb{R}^{2L} \times \mathbb{R}^N} f(\hat{\mathbf{x}}, \hat{\mathbf{F}}) \triangleq \mathbb{1}^T \hat{\mathbf{x}} + \frac{1}{2} \rho \hat{\mathbf{x}}^T \hat{\mathbf{x}} + \frac{1}{2} \rho \hat{\mathbf{F}}^T \hat{\mathbf{F}}$$

$$
\begin{aligned}
&+ \gamma_1 \sum_{i\in\mathcal{V}} \varphi\Big(N - 1 - \sum_{j\in\mathcal{N}_i} \hat{x}_{(i,j)}\Big) \\
&+ \gamma_2 \sum_{\{i,j\}\in\mathcal{E}} \varphi\Big(\hat{x}_{(i,j)} + \hat{x}_{(j,i)} - N\Big) \\
&+ \frac{1}{2}\gamma_3 \sum_{\{i,j\}\in\mathcal{E}} \Big(\hat{F}_i - \hat{F}_j - \hat{x}_{(i,j)} + \hat{x}_{(j,i)}\Big)^2 \\
&+ \gamma_4 \sum_{(i,j)\in\tilde{\mathcal{E}}} \varphi\Big(|\mathcal{N}_j/\mathcal{N}_i| - \hat{x}_{(i,j)}\Big) \\
&+ \gamma_5 \sum_{(i,j)\in\tilde{\mathcal{E}}} \varphi\Big(\hat{x}_{(i,j)} - N + |\mathcal{N}_i|\Big) \\
&+ \gamma_6 \sum_{(i,j)\in\tilde{\mathcal{E}}} \varphi\Big(\hat{x}_{(i,j)} - 1 - \sum_{\substack{k\in\mathcal{N}_j/\mathcal{N}_i \\ k\neq i}} \hat{x}_{(j,k)}\Big),
\end{aligned} \tag{4.45}
$$

and

$$
\begin{aligned}
&\min_{\hat{\mathbf{B}}\in\mathbb{R}^{2L+N}} g(\hat{\mathbf{B}};\hat{\mathbf{x}}) \triangleq \mathbb{1}^T\hat{\mathbf{B}} + \frac{1}{2}\rho'\hat{\mathbf{B}}^T\hat{\mathbf{B}} \\
&+ \frac{1}{2}\gamma_1' \sum_{\{i,j\}\in\mathcal{E}} \Big(\hat{B}_{(i,j)} - \hat{B}_{(j,i)}\Big)^2 \\
&+ \gamma_2' \sum_{(i,j)\in\tilde{\mathcal{E}}} \varphi\Big(\hat{B}_{(i,j)} - \hat{x}_{(i,j)}\hat{x}_{(j,i)}\Big) \\
&+ \gamma_3' \sum_{(i,j)\in\tilde{\mathcal{E}}} \varphi\Big(\hat{x}_{(i,j)} + \hat{x}_{(j,i)} - 1 - \hat{B}_{(i,j)}\Big) \\
&+ \gamma_4' \sum_{(i,j)\in\tilde{\mathcal{E}}} \varphi\Big(\hat{B}_{(i,j)} - \hat{x}_{(i,j)} - \sum_{\substack{h\in\mathcal{N}_i \\ h\neq j}} (\hat{B}_{(i,h)} - \hat{x}_{(i,h)})\Big) \\
&+ \gamma_5' \sum_{(i,j)\in\tilde{\mathcal{E}}} \varphi\Big(\sum_{\substack{h\in\mathcal{N}_i/\mathcal{N}_j \\ h\neq j}} (\hat{x}_{(h,i)} - 1) - \hat{B}_i + \hat{B}_{(i,j)} - \hat{x}_{(i,j)}\Big) \\
&+ \gamma_6' \sum_{i\in\mathcal{V}} \varphi\Big(\hat{B}_i - \sum_{j\in\mathcal{N}_i} \sum_{\substack{h\in\mathcal{N}_i \\ h\neq j}} \hat{x}_{(i,j)}\hat{x}_{(i,h)}\Big) \\
&+ \gamma_7' \sum_{i\in\mathcal{V}} \varphi\Big(\sum_{j\in\mathcal{N}_i} \hat{x}_{(j,i)} - |\mathcal{N}_i| - \hat{B}_i\Big) \\
&+ \gamma_8' \sum_{i\in\mathcal{V}} \varphi\Big(\hat{B}_i - \sum_{j\in\mathcal{N}_i} (\hat{B}_{(i,j)} - \hat{x}_{(i,j)})\Big)
\end{aligned}
$$

$$+\gamma'_9 \sum_{(i,j)\in\tilde{\mathcal{E}}} \varphi\Big(2\hat{B}_{(i,j)} - \hat{B}_i - 2\hat{x}_{(i,j)}\Big)$$
$$+\gamma'_{10} \sum_{(i,j)\in\tilde{\mathcal{E}}} \varphi\Big(\hat{B}_i - 2\hat{B}_{(i,j)} - (N - \hat{x}_{(i,j)})^2 + 2\hat{x}_{(i,j)}\Big), \tag{4.46}$$

where $f : \mathbb{R}^{2L} \times \mathbb{R}^N \to \mathbb{R}$ is the objective function of the first optimization problem, $\mathbb{1}$ is the all-one column vector, $\rho > 0$, $\gamma_1, \gamma_2, \ldots, \gamma_6 > 0$, $g : \mathbb{R}^{2L+N} \to \mathbb{R}$ is the objective function of the second optimization problem, the semicolon in $g(\hat{\mathbf{B}}; \hat{\mathbf{x}})$ indicates that $\hat{\mathbf{x}}$ is to be treated as a constant parameter, $\rho' > 0$, $\gamma'_1, \gamma'_2, \ldots, \gamma'_{10} > 0$, and $\varphi : \mathbb{R} \to \mathbb{R}$ is a barrier-like function defined as

$$\varphi(z) = \begin{cases} \frac{1}{2}z^2, & z \geq 0, \\ 0, & z < 0. \end{cases}$$

The first and second optimization problems (4.45) and (4.46) are associated with the first and second parts of the distributed algorithm, respectively. These two problems have the following interpretation: the first term $\mathbb{1}^T\hat{\mathbf{x}}$ in (4.45) and the first term $\mathbb{1}^T\hat{\mathbf{B}}$ in (4.46) are motivated by our intution that the $x_{(i,j)}$'s, $B_{(i,j)}$'s, and B'_i's in most graphs are closer to their lower bounds provided in Theorems 9 and 10, than to their upper bounds provided in the theorems. The second term $\frac{1}{2}\rho\hat{\mathbf{x}}^T\hat{\mathbf{x}} + \frac{1}{2}\rho\hat{\mathbf{F}}^T\hat{\mathbf{F}}$ in (4.45) and the second term $\frac{1}{2}\rho'\hat{\mathbf{B}}^T\hat{\mathbf{B}}$ in (4.46) are quadratic regularization terms intended to ensure that the optimization problems are strongly convex and thus have unique solutions. Hence, both ρ and ρ' are meant to be small. The rest of the terms in (4.45) and in (4.46) are inspired by the barrier method of converting constrained optimization problems into unconstrained ones. In particular, the 6 terms in (4.45) model the constraint set Θ_1, while the 10 terms in (4.46) model the constraint set $\Theta_2(\hat{\mathbf{x}})$. Therefore, the γ_ℓ's and γ'_ℓ's are meant to be large.

As it follows from the above, $f(\hat{\mathbf{x}}, \hat{\mathbf{F}})$ in (4.45) assumes a small value if $(\hat{\mathbf{x}}, \hat{\mathbf{F}}) \in \Theta_1$, assumes a large value if $(\hat{\mathbf{x}}, \hat{\mathbf{F}}) \notin \Theta_1$, is strongly convex in $(\hat{\mathbf{x}}, \hat{\mathbf{F}})$, and has a unique

minimizer $(\mathbf{x}^*, \mathbf{F}^*) \in \Theta_1$ for sufficiently small ρ and large γ_ℓ's. Likewise, $g(\hat{\mathbf{B}}; \hat{\mathbf{x}})$ in (4.46) assumes a small value if $\hat{\mathbf{B}} \in \Theta_2(\hat{\mathbf{x}})$, assumes a large value if $\hat{\mathbf{B}} \notin \Theta_2(\hat{\mathbf{x}})$, is strongly convex in $\hat{\mathbf{B}}$ for each fixed $\hat{\mathbf{x}}$, and has a unique minimizer $\mathbf{B}^*(\hat{\mathbf{x}}) \in \Theta_2(\hat{\mathbf{x}})$ for sufficiently small ρ' and large γ'_ℓ's. Therefore, by having the N nodes cooperatively execute a continuous-time gradient-descent algorithm

$$\begin{bmatrix} \dot{\hat{\mathbf{x}}}(t) \\ \dot{\hat{\mathbf{F}}}(t) \end{bmatrix} = -\varepsilon \begin{bmatrix} \nabla_{\hat{\mathbf{x}}} f(\hat{\mathbf{x}}(t), \hat{\mathbf{F}}(t)) \\ \nabla_{\hat{\mathbf{F}}} f(\hat{\mathbf{x}}(t), \hat{\mathbf{F}}(t)) \end{bmatrix}, \tag{4.47}$$

$$\dot{\hat{\mathbf{B}}}(t) = -\varepsilon' \nabla_{\hat{\mathbf{B}}} g(\hat{\mathbf{B}}(t); \hat{\mathbf{x}}(t)), \tag{4.48}$$

where $\varepsilon > 0$ and $\varepsilon' > 0$, the estimates $(\hat{\mathbf{x}}(t), \hat{\mathbf{F}}(t), \hat{\mathbf{B}}(t))$ are guaranteed to asymptotically converge to the point $(\mathbf{x}^*, \mathbf{F}^*, \mathbf{B}^*(\mathbf{x}^*))$ in the set Θ. We note that because the 16 constraints are distributed in nature, algorithm (4.47)–(4.48) is distributedly implementable, requiring only communications among neighboring nodes. Indeed, this feature can be seen by rewriting algorithm (4.47)–(4.48) in an element-wise form as follows: first, for each $(i, j) \in \tilde{\mathcal{E}}$,

$$\begin{aligned} \dot{\hat{x}}_{(i,j)}(t) = -\varepsilon \Big[& 1 + \rho \hat{x}_{(i,j)}(t) - \gamma_1 \varphi' \Big(N - 1 - \sum_{k \in \mathcal{N}_i} \hat{x}_{(i,k)}(t) \Big) \\ & + \gamma_2 \varphi'(\hat{x}_{(i,j)}(t) + \hat{x}_{(j,i)}(t) - N) - \gamma_3 (\hat{F}_i(t) - \hat{F}_j(t) - \hat{x}_{(i,j)}(t) + \hat{x}_{(j,i)}(t)) \\ & - \gamma_4 \varphi'(|\mathcal{N}_j / \mathcal{N}_i| - \hat{x}_{(i,j)}(t)) + \gamma_5 \varphi'(\hat{x}_{(i,j)}(t) - N + |\mathcal{N}_i|) \\ & + \gamma_6 \varphi' \Big(\hat{x}_{(i,j)}(t) - 1 - \sum_{k \in \mathcal{N}_j / (\mathcal{N}_i \cup \{i\})} \hat{x}_{(j,k)}(t) \Big) \\ & - \gamma_6 \sum_{h \in \mathcal{N}_i / (\mathcal{N}_j \cup \{j\})} \varphi' \Big(\hat{x}_{(h,i)}(t) - 1 - \sum_{k \in \mathcal{N}_i / (\mathcal{N}_h \cup \{h\})} \hat{x}_{(i,k)}(t) \Big) \Big], \end{aligned} \tag{4.49}$$

where

$$\varphi'(z) = \begin{cases} z, & z \geq 0, \\ 0, & z < 0. \end{cases} \tag{4.50}$$

Second, for each $i \in \mathcal{V}$,

$$\dot{\hat{F}}_i(t) = -\varepsilon\Big[\rho\hat{F}_i(t) + \gamma_3 \sum_{j\in\mathcal{N}_i}(\hat{F}_i(t) - \hat{F}_j(t) - \hat{x}_{(i,j)}(t) + \hat{x}_{(j,i)}(t))\Big]. \tag{4.51}$$

Third, for each $(i,j) \in \tilde{\mathcal{E}}$,

$$\begin{aligned}
\dot{\hat{B}}_{(i,j)} = -\varepsilon'\Big[&\rho'\hat{B}_{(i,j)} + \gamma_1'\Big(\hat{B}_{(i,j)} - \hat{B}_{(j,i)}\Big) + \gamma_2'\varphi'\Big(\hat{B}_{(i,j)} - \hat{x}_{(i,j)}\hat{x}_{(j,i)}\Big) \\
&- \gamma_3'\varphi'\Big(\hat{x}_{(i,j)} + \hat{x}_{(j,i)} - 1 - \hat{B}_{(i,j)}\Big) \\
&+ \gamma_4'\varphi'\Big(\hat{B}_{(i,j)} - \hat{x}_{(i,j)} - \sum_{\substack{h\in\mathcal{N}_i \\ h\neq j}}(\hat{B}_{(i,h)} - \hat{x}_{(i,h)})\Big) \\
&- \gamma_4'\sum_{\substack{h\in\mathcal{N}_i \\ h\neq j}}\varphi'\Big(\hat{B}_{(i,h)} - \hat{x}_{(i,h)} - \sum_{\substack{k\in\mathcal{N}_i \\ k\neq h}}(\hat{B}_{(i,k)} - \hat{x}_{(i,k)})\Big) \\
&+ \gamma_5'\varphi'\Big(\sum_{\substack{h\in\mathcal{N}_i/\mathcal{N}_j \\ h\neq j}}(\hat{x}_{(h,i)} - 1) - \hat{B}_i + \hat{B}_{(i,j)} - \hat{x}_{(i,j)}\Big) \\
&- \gamma_8'\varphi'\Big(\hat{B}_i - \sum_{k\in\mathcal{N}_i}(\hat{B}_{(i,k)} - \hat{x}_{(i,k)})\Big) + 2\gamma_9'\varphi'\Big(2\hat{B}_{(i,j)} - \hat{B}_i - 2\hat{x}_{(i,j)}\Big) \\
&- 2\gamma_{10}'\varphi'\Big(\hat{B}_i - 2\hat{B}_{(i,j)} - (N - \hat{x}_{(i,j)})^2 + 2\hat{x}_{(i,j)}\Big)\Big].
\end{aligned} \tag{4.52}$$

Finally, for each $i \in \mathcal{V}$,

$$\begin{aligned}
\dot{\hat{B}}_i = -\varepsilon'\Big[&\rho'\hat{B}_i - \gamma_5'\sum_{j\in\mathcal{N}_i}\varphi'\Big(\sum_{\substack{h\in\mathcal{N}_i/\mathcal{N}_j \\ h\neq j}}(\hat{x}_{(h,i)} - 1) - \hat{B}_i + \hat{B}_{(i,j)} - \hat{x}_{(i,j)}\Big) \\
&+ \gamma_6'\varphi'\Big(\hat{B}_i - \sum_{j\in\mathcal{N}_i}\sum_{\substack{h\in\mathcal{N}_i \\ h\neq j}}\hat{x}_{(i,j)}\hat{x}_{(i,h)}\Big) - \gamma_7'\varphi'\Big(\sum_{j\in\mathcal{N}_i}\hat{x}_{(j,i)} - |\mathcal{N}_i| - \hat{B}_i\Big)
\end{aligned}$$

$$+\gamma_8'\varphi'\Big(\hat{B}_i-\sum_{j\in\mathcal{N}_i}(\hat{B}_{(i,j)}-\hat{x}_{(i,j)})\Big)-\gamma_9'\sum_{j\in\mathcal{N}_i}\varphi'\Big(2\hat{B}_{(i,j)}-\hat{B}_i-2\hat{x}_{(i,j)}\Big)$$
$$+\gamma_{10}'\sum_{j\in\mathcal{N}_i}\varphi'\Big(\hat{B}_i-2\hat{B}_{(i,j)}-(N-\hat{x}_{(i,j)})^2+2\hat{x}_{(i,j)}\Big)\Big]. \tag{4.53}$$

Examining (4.49)–(4.53), we see that algorithm (4.47)–(4.48) is scalable with memory and communication complexities of $O(|\mathcal{N}_i|)$. We also note that although the point $(\mathbf{x}^*, \mathbf{F}^*, \mathbf{B}^*(\mathbf{x}^*))$ and aggregated variables $(\mathbf{x}, \mathbf{F}, \mathbf{B})$ are both in the set Θ, we presently do not have analytical bounds on the distance between them.

4.5 Performance Evaluation

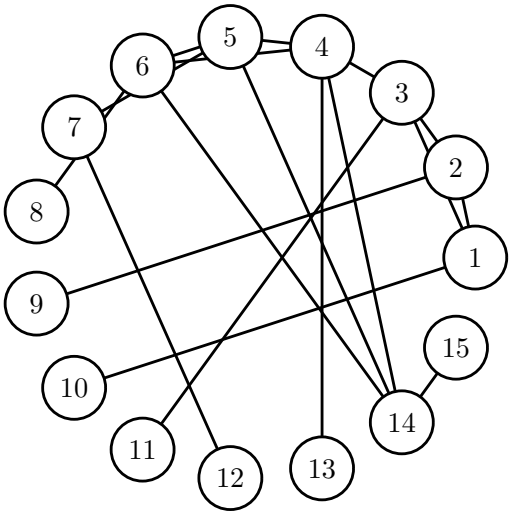
In this section, we evaluate the performance of algorithm (4.47)–(4.48) via two sets of simulations. Section 4.5.1 presents the first set, while Sections 4.5.2 and 4.5.3 present the second.

4.5.1 First Set of Simulation

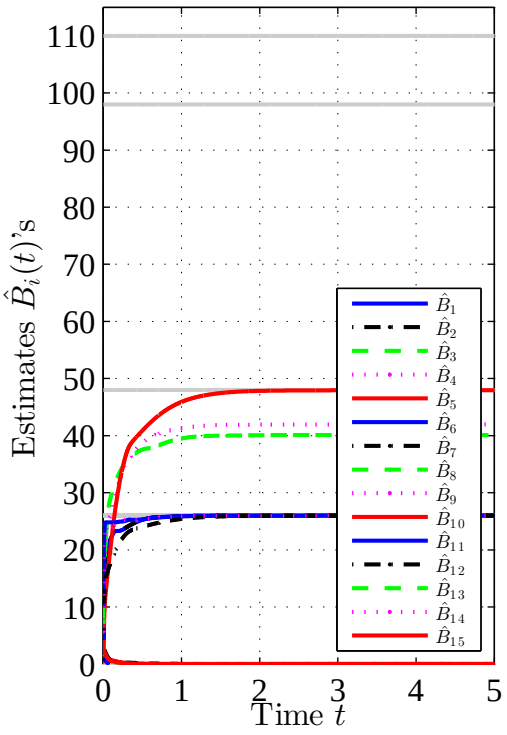
For the first set of simulation, we illustrate the performance of algorithm (4.47)–(4.48) on a 15-node graph. Figure 4.1(a) displays the graph including the node indices. Figures 4.1(b) and 4.1(c) display as functions of time t the node and edge betweenness B_i''s and $B_{(i,j)}$'s using gray lines, and their estimates $\hat{B}_i(t)$'s and $\hat{B}_{(i,j)}(t)$'s using color curves. Observe from the figures that, as expected, the $\hat{B}_i(t)$'s and $\hat{B}_{(i,j)}(t)$'s converge to some steady-state values as $t \to \infty$. Moreover, these steady-state values coincide with the B_i''s and $B_{(i,j)}$'s for most, but not all, of the nodes and edges.

4.5.2 Second Set of Simulation: Evaluation Settings

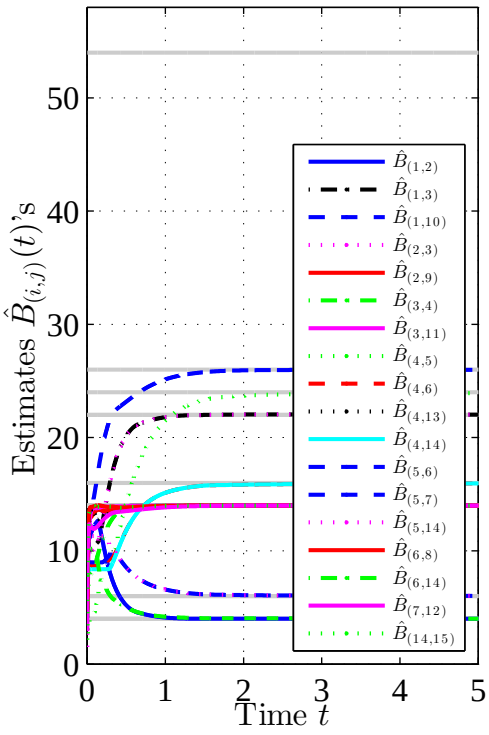
For the second set of simulation, we evaluate the performance of algorithm (4.47)–(4.48) on random geometric graphs. Specifically, we consider four values of N from the



(a) Node indices.



(b) Estimates $\hat{B}_i(t)$'s.



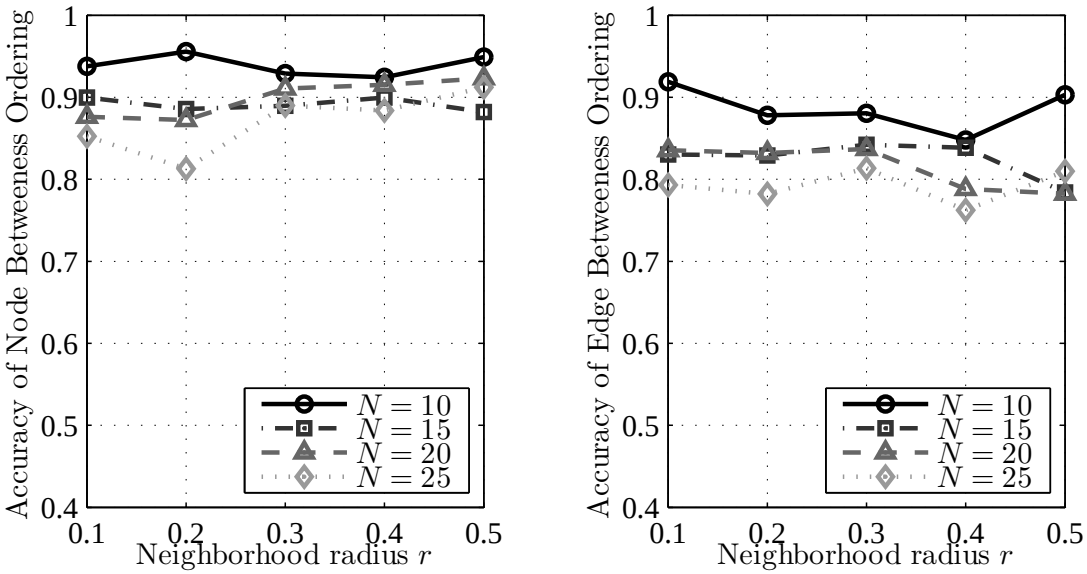
(c) Estimates $\hat{B}_{(i,j)}(t)$'s.

Figure 4.1: An illustration of the performance of algorithm (4.47)–(4.48) in estimating betweenness on a 15-node graph.

set $\{10, 15, 20, 25\}$ and five values of neighborhood radius r from the set $\{0.1, 0.2, 0.3, 0.4, 0.5\}$. For each of the 20 combinations of (N, r), we generate 10 different scenarios. For each scenario k, we place N nodes randomly and equiprobably on the unit square $[0, 1] \times [0, 1]$, regard any two nodes as neighbors if their distance does not exceed r, and repeat the process until the resulting graph is connected. We then simulate algorithm (4.47)–(4.48) with $\varepsilon = \varepsilon' = 5 \times 10^{-5}$, $\rho = \rho' = 0.001$, $\gamma_\ell = 10^6$ $\forall \ell \in \{1, 2, \ldots, 6\}$, and $\gamma'_\ell = 10^6$ $\forall \ell \in \{1, 2, \ldots, 10\}$ and record the resulting $\hat{x}_{(i,j)}(T)$ $\forall (i, j) \in \tilde{\mathcal{E}}$, $\hat{F}_i(T)$ $\forall i \in \mathcal{V}$, $\hat{B}_{(i,j)}(T)$ $\forall (i, j) \in \tilde{\mathcal{E}}$, and $\hat{B}_i(T)$ $\forall i \in \mathcal{V}$, where T is the simulation duration. Upon completion, we compare the N node betweenness estimates $\hat{B}_1(T), \hat{B}_2(T), \ldots, \hat{B}_N(T)$ with their true values $B'_1, B'_2, \ldots, B'_N$ computed from (4.1). To facilitate the comparison, we introduce a measure called *Accuracy of Ordering*, denoted as $\mathrm{AO}(N, r, k)$ (i.e., for scenario k of combination (N, r)) and defined as

$$\mathrm{AO}(N, r, k) = \frac{2}{N(N-1)} \sum_{i \in \mathcal{V}} \sum_{\substack{j \in \mathcal{V} \\ j > i}} \mathcal{O}(\hat{B}_i(T), \hat{B}_j(T), B'_i, B'_j),$$

where $\mathcal{O}(a_1, a_2, b_1, b_2)$ is a binary-valued function that returns 1 if $a_1 \leq a_2$ and $b_1 \leq b_2$ or if $a_1 \geq a_2$ and $b_1 \geq b_2$ (i.e., (a_1, a_2) and (b_1, b_2) have the same order), and returns 0 otherwise (i.e., different order). Thus, the Accuracy of Ordering measures how well the node betweenness estimates match up with their true values in terms of ordering, takes the maximum value of 1 when the order is completely preserved, and takes the minimum value of 0 when the order is completely reversed. Upon calculating $\mathrm{AO}(N, r, k)$ for each scenario $k \in \{1, 2, \ldots, 10\}$, we record their average $\frac{1}{10} \sum_{k=1}^{10} \mathrm{AO}(N, r, k)$ as $\mathrm{AO}(N, r)$. In a similar fashion, we use the measure $\mathrm{AO}(N, r, k)$ to characterize how well the $2L$ edge betweenness estimates $\hat{B}_{(i,j)}(T)$ $\forall (i, j) \in \tilde{\mathcal{E}}$ match up with their true values $B_{(i,j)}$ $\forall (i, j) \in \tilde{\mathcal{E}}$ computed from (4.2) and (4.21), and record the average also as $\mathrm{AO}(N, r)$.



(a) Node betweenness.

(b) Edge betweenness.

Figure 4.2: Performance of algorithm (4.47)–(4.48) on random geometric graphs as measured by the Accuracy of Ordering.

4.5.3 Second Set of Simulation: Evaluation Results

Figure 4.2 displays the evaluation results. Specifically, Figures 4.2(a) and 4.2(b) represent, respectively, the Accuracy of Ordering $\mathrm{AO}(N, r)$ for node and edge betweenness as functions of the number of nodes N and the neighborhood radius r. Analyzing these figures, we see that algorithm (4.47)–(4.48) performs fairly well, achieving Accuracy of Ordering scores of 0.9000 for node betweenness and 0.8294 for edge betweenness, in all the scenarios considered.

4.6 Conclusion

In this chapter, we have developed a scalable distributed algorithm that enables nodes in a general graph to cooperatively estimate their individual betweenness and the betweenness of edges incident on them with only local interaction and without any centralized coordination, nor high memory usages. To arrive at this algorithm, we

have exploited various local properties of shortest paths and considered an unconstrained distributed optimization problem. We have also shown via simulation on a number of random geometric graphs that the algorithm is fairly accurate in terms of ordering, but has room for improvement. Therefore, possible future research directions include deriving additional constraints that decrease the size of the feasible set Θ and incorporating them into the algorithm to increase its estimation accuracy.

Chapter 5

Distributed Estimation of Closeness Centrality

5.1 Introduction

Closeness centrality is one of the most fundamental centrality measures [3, 12, 13] that quantifies how centrally located a node is, within a network, based on its total distances to all other nodes. Specifically, given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, \ldots, N\}$ denotes the set of N nodes and $\mathcal{E}$ the set of edges, the *closeness centrality* or simply *closeness* C_i of a node $i \in \mathcal{V}$ is defined in (1.2) [12, 13] as

$$C_i \triangleq \frac{N-1}{\sum_{j \in \mathcal{V}} d_{ij}},$$

where $d_{ij} = d_{ji}$ is the distance between nodes i and j. As can be seen from (1.2), the larger C_i, the closer node i is, on average, to all other nodes in the graph $\mathcal{G}$. Thus, this concept of closeness is useful and has been applied to a number of areas, including epidemiology [14], social networks [15], and power systems [16, 17].

Although the concept of closeness is useful, its computation in a decentralized

setting has received little attention in the literature. Indeed, majority of existing work in the area of centrality measures focuses on introducing new measures as well as applying existing ones to various applications. To our understanding, [23] and [121] are the only two prior work on distributed computation of closeness. In particular, [23] proposes a distributed algorithm for computing closeness which is applicable to general graphs but suffers from very high memory requirement and algorithm complexity. In contrast, [121] proposes a distributed algorithm for computing both closeness and one of its variants called exponential closeness, which is simple and scalable but is applicable only to tree graphs.

Motivated by the above considerations, in this chapter we address the problem of distributed computation of closeness on general graphs. To do so, we begin with the derivation of a set of linear inequality and equality constraints, which characterize closeness in place of its original definition, and which are distributed in nature so that neighboring nodes can check whether and how well they are satisfied. Using these constraints, we subsequently develop a scalable distributed algorithm that enables every node $i \in \mathcal{V}$ to determine an estimate $\hat{C}_i$ of its own closeness C_i with only local interaction and without any centralized coordination, nor high memory usages. As will be shown, this distributed algorithm is actually a gradient algorithm that solves a regularized linear program whose constraints are augmented to its objective function as barriers. Finally, we simulate the algorithm on random geometric, Erdős-Rényi, and Barabási-Albert graphs [3]. The results show that the estimates $\hat{C}_1, \hat{C}_2, \ldots, \hat{C}_N$ are 91% accurate in terms of ordering, compared to the ordering of their true values $C_1, C_2, \ldots, C_N$.

The outline of this chapter is as follows: Section 5.2 formulates the problem addressed. Section 5.3 describes the solution approach, on which Sections 5.4 and 5.5 are based. Section 5.4 derives the set of distributed constraints that characterize closeness. Section 5.5 derives the distributed algorithm for estimating closeness. Sec-

tion 5.6 evaluates its performance via extensive simulation. Finally, Section 5.7 concludes the chapter.

5.2 Problem Formulation

Consider a network modeled as an undirected, unweighted and connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, \ldots, N\}$ denotes the set of $N \geq 2$ nodes and $\mathcal{E} \subset \{\{i, j\} : i, j \in \mathcal{V}, i \neq j\}$ denotes the set of L edges. Any two nodes $i, j \in \mathcal{V}$ are neighbors and can communicate if and only if $\{i, j\} \in \mathcal{E}$. The set of neighbors of each node $i \in \mathcal{V}$ is denoted as $\mathcal{N}_i = \{j \in \mathcal{V} : \{i, j\} \in \mathcal{E}\}$, and the communications are assumed to be delay- and error-free, with no quantization.

Suppose each node $i \in \mathcal{V}$ knows only the value of N and its set $\mathcal{N}_i$ of neighbors. Also suppose each node $i \in \mathcal{V}$ is willing to share its knowledge of $\mathcal{N}_i$ with every neighbor $j \in \mathcal{N}_i$, but not with others due perhaps to security and privacy reasons. Yet, despite having only such local information about the graph $\mathcal{G}$, suppose every node $i \in \mathcal{V}$ wants to determine how centrally located it is, as measured by its closeness C_i defined in (1.2).

Given the above, the goal of this chapter is to develop a scalable distributed algorithm that enables every node $i \in \mathcal{V}$ to estimate its closeness C_i with a good accuracy and without using a significant amount of memory.

5.3 Solution Approach

In this and the next two sections, we design a distributed algorithm that achieves the aforementioned goal. Because the design involves multiple steps, for clarity we describe in this section the ideas behind the design and provide in the next two sections the technical details.

Reconsider the graph $\mathcal{G}$ and suppose we associate with each node $i \in \mathcal{V}$ an n_i-

dimensional vector $z_i \in \mathbb{R}^{n_i}$. These N vectors $z_1, z_2, \ldots, z_N$ are intended to have meaning that depends on the graph $\mathcal{G}$. For example, for each $i \in \mathcal{V}$ we could let $n_i = 2$ and let the first entry of z_i be the number of shortest paths in $\mathcal{G}$ that go through node i, and the second entry of z_i be the number of shortest paths in $\mathcal{G}$ that begin at node i. We note that this is just an example and, for the moment, we will leave the definition of the N vectors z_i's open. Next, let $\mathbf{z} = (z_1, z_2, \ldots, z_N) \in \mathbb{R}^n$ denote the vector obtained by stacking the N vectors z_i's, where $n = \sum_{i \in \mathcal{V}} n_i$. Suppose the vector $\mathbf{z}$ can be *defined* in such a way that it has the following four properties:

Property 1. For each $i \in \mathcal{V}$, there exists a function $\phi_i : \mathbb{R}^{n_i} \to \mathbb{R}$ known to node i, such that

$$C_i = \phi_i(z_i), \quad \forall i \in \mathcal{V}. \tag{5.1}$$

Property 2. For each $i \in \mathcal{V}$, there exists a set $Z_i \subset \mathbb{R}^{n_i}$ known to node i, such that

$$z_i \in Z_i, \quad \forall i \in \mathcal{V}. \tag{5.2}$$

Property 3. For each $i \in \mathcal{V}$ and $j \in \mathcal{N}_i$, there exists a set $Z_{(i,j)} \subset \mathbb{R}^{n_i+n_j}$ known to nodes i and j, such that

$$(z_i, z_j) \in Z_{(i,j)}, \quad \forall i \in \mathcal{V},\ \forall j \in \mathcal{N}_i. \tag{5.3}$$

Property 4. If a vector $\tilde{\mathbf{z}} = (\tilde{z}_1, \tilde{z}_2, \ldots, \tilde{z}_N) \in \mathbb{R}^n$ satisfies

$$\tilde{z}_i \in Z_i, \quad \forall i \in \mathcal{V}, \tag{5.4}$$

$$(\tilde{z}_i, \tilde{z}_j) \in Z_{(i,j)}, \quad \forall i \in \mathcal{V},\ \forall j \in \mathcal{N}_i, \tag{5.5}$$

then $\tilde{\mathbf{z}} = \mathbf{z}$.

Property 1 suggests that if each node $i \in \mathcal{V}$ is able to *learn* the value of z_i, it could calculate its own closeness C_i using (5.1) because it knows the function ϕ_i. To learn the value of z_i, suppose each node $i \in \mathcal{V}$ maintains in its local memory a vector $\hat{z}_i(t) \in \mathbb{R}^{n_i}$, which represents its estimate of the unknown z_i at time $t \in [0, \infty)$ (assuming continuous-time). As for how does each node $i \in \mathcal{V}$ update its estimate $\hat{z}_i(t)$ so that $\hat{z}_i(t)$ gradually approaches z_i, the nodes could make use of constraints (5.2) and (5.3) in Properties 2 and 3. That is, the nodes could cooperate with their neighbors to jointly update their estimates so that $\hat{z}_i(t)$ gradually approaches the set Z_i for each $i \in \mathcal{V}$, and $(\hat{z}_i(t), \hat{z}_j(t))$ gradually approaches the set $Z_{(i,j)}$ for each $i \in \mathcal{V}$ and $j \in \mathcal{N}_i$. Notice that such an update is implementable because the set Z_i is known to node i for each $i \in \mathcal{V}$, and the set $Z_{(i,j)}$ is known to nodes i and j for each $i \in \mathcal{V}$ and $j \in \mathcal{N}_i$. In other words, constraints (5.2) and (5.3) are distributed in nature, allowing neighboring nodes to check whether and how well they are satisfied. Lastly, Property 4 suggests that if the nodes are able to drive $\hat{z}_i(t)$ into the set Z_i for each $i \in \mathcal{V}$, and $(\hat{z}_i(t), \hat{z}_j(t))$ into the set $Z_{(i,j)}$ for each $i \in \mathcal{V}$ and $j \in \mathcal{N}_i$, the estimates $\hat{z}_i(t)$'s would be equal to the unknowns z_i's because the property implies that there is exactly one point in $\mathbb{R}^n$ satisfying (5.4) and (5.5), and that point is $\mathbf{z}$.

As it follows from the above, if we are able to define the vector $\mathbf{z}$ so that it has Properties 1–4, and come up with a distributed algorithm that drives the estimates $\hat{z}_i(t)$'s into the sets Z_i's and $Z_{(i,j)}$'s, the closeness estimation problem in Section 5.2 would be solved. This describes the ideas behind our design. In the next two sections, we provide the details.

5.4 Distributed Characterization of Closeness

In this section, we show that it is possible to define the vector $\mathbf{z}$ so that it has Properties 1–4.

To set the stage, consider the following notations. First, although $\mathcal{G}$ is undirected, let us associate with each edge $\{i,j\} \in \mathcal{E}$ a pair of directed edges denoted as (i,j) and (j,i) (i.e., braces are for undirected edges, while parentheses are for directed ones). Moreover, let $\tilde{\mathcal{E}}$ denote the set of $2L$ directed edges. Furthermore, since we often need to refer to the reciprocal of C_i, for convenience we define here the *farness* F_i of a node $i \in \mathcal{V}$ in (4.20) can be rewritten as

$$F_i \triangleq \sum_{j \in \mathcal{V}} d_{ij},$$

so that $F_i = \frac{N-1}{C_i}$. Similarly, we often need to refer to d_{ij} with j treated as *fixed*. Hence, whenever we wish to emphasize that j is fixed we write d_{ij} as $d_i^{(j)}$.

Observe that in order for each node $i \in \mathcal{V}$ to determine its closeness C_i from (1.2) or, equivalently, its farness F_i from (4.20), it must know the N distances d_{ij} $\forall j \in \mathcal{V}$. Unfortunately, such distances are not available to node i, except for d_{ii} which is 0, and d_{ij} $\forall j \in \mathcal{N}_i$ which are 1. However, it might be possible that the N^2 distances d_{ij} $\forall i,j \in \mathcal{V}$ are *constrained in some ways*, which the nodes could exploit in their attempt to determine the d_{ij}'s they need. Below, we show that this is indeed the case by deriving a set of constraints involving the d_{ij}'s which turns out to be distributed in nature (in the sense of Properties 2 and 3).

To begin, consider a node $s \in \mathcal{V}$, which is meant to be *fixed*, and the set of all shortest paths ending at node s, which make node s act like a "sink" (hence the symbol s). With this in mind, for each directed edge $(i,j) \in \tilde{\mathcal{E}}$, let

$$x_{(i,j)}^{(s)} \triangleq \begin{cases} 1, & \text{if there exists a shortest path from node } i \\ & \text{to node } s \text{ that goes through edge } (i,j), \\ 0, & \text{otherwise.} \end{cases}$$

as shown in (4.3), defines $2L$ binary variables $x^{(s)}_{(i,j)}$ $\forall (i,j) \in \tilde{\mathcal{E}}$, each of which has a graph-theoretic meaning. These variables, together with the N distances $d^{(s)}_i$ $\forall i \in \mathcal{V}$, form a set of $2L+N$ variables having the following properties:

Theorem 11. *The $2L+N$ variables $x^{(s)}_{(i,j)}$ $\forall (i,j) \in \tilde{\mathcal{E}}$ and $d^{(s)}_i$ $\forall i \in \mathcal{V}$ satisfy*

$$\begin{aligned}
&\sum_{j \in \mathcal{N}_s} x^{(s)}_{(s,j)} = 0, \\
&\sum_{j \in \mathcal{N}_i} x^{(s)}_{(i,j)} \geq 1, \quad \forall i \neq s, \\
&x^{(s)}_{(i,j)} + x^{(s)}_{(j,i)} \leq 1, \quad \forall \{i,j\} \in \mathcal{E}, \\
&d^{(s)}_i - d^{(s)}_j = x^{(s)}_{(i,j)} - x^{(s)}_{(j,i)}, \quad \forall \{i,j\} \in \mathcal{E}, \\
&d^{(s)}_s = 0.
\end{aligned}$$

Proof. See Lemma 5 in Chapter4. □

An immediate implication of the above observation is that although the nodes do not know the values of the variables $x^{(s)}_{(i,j)}$ $\forall (i,j) \in \tilde{\mathcal{E}}$ and $d^{(s)}_i$ $\forall i \in \mathcal{V}$, they could locally maintain *estimates* of these variables, denoted as $\hat{x}^{(s)}_{(i,j)} \in \{0,1\}$ $\forall (i,j) \in \tilde{\mathcal{E}}$ and $\hat{d}^{(s)}_i \in \mathbb{R}$ $\forall i \in \mathcal{V}$. By repeatedly exchanging latest values of their estimates with their neighbors', and repeatedly checking how well their estimates satisfy constraints (4.8)–(4.12), the nodes might be able to drive their estimates toward satisfying all the constraints simultaneously, i.e.,

$$\sum_{j \in \mathcal{N}_s} \hat{x}^{(s)}_{(s,j)} = 0, \tag{5.6}$$

$$\sum_{j \in \mathcal{N}_i} \hat{x}^{(s)}_{(i,j)} \geq 1, \quad \forall i \neq s, \tag{5.7}$$

$$\hat{x}^{(s)}_{(i,j)} + \hat{x}^{(s)}_{(j,i)} \leq 1, \quad \forall \{i,j\} \in \mathcal{E}, \tag{5.8}$$

$$\hat{d}^{(s)}_i - \hat{d}^{(s)}_j = \hat{x}^{(s)}_{(i,j)} - \hat{x}^{(s)}_{(j,i)}, \quad \forall \{i,j\} \in \mathcal{E}, \tag{5.9}$$

$$\hat{d}_s^{(s)} = 0. \tag{5.10}$$

If the nodes are able to force their estimates to satisfy constraints (5.6)–(5.10), and if *constraints* (5.6)–(5.10) *have a unique feasible point*—which must then be $x_{(i,j)}^{(s)}$ $\forall (i,j) \in \tilde{\mathcal{E}}$ and $d_i^{(s)}$ $\forall i \in \mathcal{V}$ since they must satisfy (4.8)–(4.12)—each node $i \in \mathcal{V}$ would have determined the d_{ij}'s it needs in order to determine its own C_i from (1.2) or F_i from (4.20) (see Property 1).

The following key theorem shows that constraints (5.6)–(5.10) indeed have a unique feasible point in the mixed-integer space $\{0,1\}^{2L} \times \mathbb{R}^N$ (see Property 4):

Theorem 12. *There exist unique* $\hat{x}_{(i,j)}^{(s)} \in \{0,1\}$ $\forall (i,j) \in \tilde{\mathcal{E}}$ *and* $\hat{d}_i^{(s)} \in \mathbb{R}$ $\forall i \in \mathcal{V}$, *given by* $\hat{x}_{(i,j)}^{(s)} = x_{(i,j)}^{(s)}$ *and* $\hat{d}_i^{(s)} = d_i^{(s)}$, *which satisfy* (5.6)–(5.10).

Proof. See Section 5.4.1. □

To summarize this section, we have shown that it is possible to define the vector $\mathbf{z} = (z_1, z_2, \ldots, z_N)$ so that it has Properties 1–4. Indeed, for each $i \in \mathcal{V}$, z_i is the vector formed by $x_{(i,j)}^{(s)}$ $\forall j \in \mathcal{N}_i$ $\forall s \in \mathcal{V}$ and $d_i^{(s)}$ $\forall s \in \mathcal{V}$. In addition, the set Z_i is defined by (4.8), (4.9), and (4.12), while the set $Z_{(i,j)}$ is defined by (4.10) and (4.11).

5.4.1 Proof of Theorem 12

To prove Theorem 12, first consider the following lemma:

Lemma 7. *If* $\hat{x}_{(i,j)}^{(s)} \in \{0,1\}$ $\forall (i,j) \in \tilde{\mathcal{E}}$ *and* $\hat{d}_i^{(s)} \in \mathbb{R}$ $\forall i \in \mathcal{V}$ *satisfy* (5.6)–(5.10), *then:*

(a. $\hat{x}_{(i,j)}^{(s)} \in \{0,1\}$ *and* $\hat{x}_{(j,i)}^{(s)} \in \{0,1\}$ *cannot be both* 1 $\forall \{i,j\} \in \mathcal{E}$,

(b. $\hat{d}_i^{(s)} - \hat{d}_j^{(s)} \in \{-1,0,1\}$ $\forall \{i,j\} \in \mathcal{E}$,

(c. $\forall i \neq s$, $\exists j \in \mathcal{N}_i$ *such that* $\hat{d}_i^{(s)} - \hat{d}_j^{(s)} = 1$.

Proof. Statement (a) follows from (4.3) and (5.8), whereas statement (b) follows from (a) and (5.9). To establish (c), note from (4.3) and (5.7) that $\forall i \neq s$, $\exists j \in \mathcal{N}_i$ such that $\hat{x}^{(s)}_{(i,j)} = 1$. Statement (c) is then a result of (a) and (5.9). □

Next, let $D = \max_{j \in \mathcal{V}} d^{(s)}_j$. Since $N \geq 2$, $D > 0$. Since $\mathcal{G}$ is connected, $D < \infty$. For each $\ell \in \{0, 1, \ldots, D\}$, let

$$\mathcal{V}_\ell = \{i \in \mathcal{V} : d^{(s)}_i = \ell\}, \tag{5.11}$$

$$m_\ell = \min_{i \in \mathcal{V}_\ell} \hat{d}^{(s)}_i, \tag{5.12}$$

$$M_\ell = \max_{i \in \mathcal{V}_\ell} \hat{d}^{(s)}_i. \tag{5.13}$$

Then, it is obvious that $\mathcal{V}$ can be partitioned into nonempty subsets $\mathcal{V}_0, \mathcal{V}_1, \ldots, \mathcal{V}_D$, and both m_ℓ and M_ℓ are well-defined since $\mathcal{V}_\ell$ is nonempty and finite.

Lemma 8. $m_0 = M_0 = 0$.

Proof. From (5.11), $\mathcal{V}_0 = \{s\}$. This, along with (5.10), (5.12), and (5.13), implies that $m_0 = M_0 = 0$. □

Lemma 9. *For each* $\ell \in \{0, 1, \ldots, D\}$, $M_\ell \leq \ell$.

Proof. By induction. From Lemma 8, $M_0 = 0$. Next, we show that $\forall \ell \in \{1, 2, \ldots, D\}$, $M_{\ell-1} \leq \ell - 1$ implies $M_\ell \leq \ell$. Suppose $M_{\ell-1} \leq \ell - 1$. Then, due to (5.13), we have $\hat{d}^{(s)}_i \leq \ell - 1$ $\forall i \in \mathcal{V}_{\ell-1}$. Now pick any $j \in \mathcal{V}_\ell$. Since $\mathcal{G}$ is connected, $\mathcal{N}_j \cap \mathcal{V}_{\ell-1} \neq \emptyset$. Let $p \in \mathcal{N}_j \cap \mathcal{V}_{\ell-1}$. Since $\hat{d}^{(s)}_i \leq \ell - 1$ $\forall i \in \mathcal{V}_{\ell-1}$, $\hat{d}^{(s)}_p \leq \ell - 1$. By Lemma 7(b), $\hat{d}^{(s)}_j \leq \hat{d}^{(s)}_p + 1 \leq \ell$. Thus, $\hat{d}^{(s)}_i \leq \ell$ $\forall i \in \mathcal{V}_\ell$. From (5.13), $M_\ell \leq \ell$, as desired. □

Lemma 10. *For each* $\ell \in \{1, 2, \ldots, D\}$, $m_{\ell-1} \leq m_\ell - 1$.

Proof. By backward induction. First, we show that the claim is true for $\ell = D$, i.e., $m_{D-1} \leq m_D - 1$. Let $j \in \arg\min_{i \in \mathcal{V}_D} \hat{d}^{(s)}_i$, so that $j \in \mathcal{V}_D$ and $\hat{d}^{(s)}_j = m_D$. Since $D \neq 0$ and since $\mathcal{V}_0 = \{s\}$, we have $j \neq s$. By Lemma 7(c), there exists $p \in \mathcal{N}_j$ such that

$\hat{d}_p^{(s)} = \hat{d}_j^{(s)} - 1$. Note that $p \notin \mathcal{V}_D$, because if $p \in \mathcal{V}_D$, then the statement $\hat{d}_p^{(s)} = \hat{d}_j^{(s)} - 1$ contradicts the statement $j \in \arg\min_{i \in \mathcal{V}_D} \hat{d}_i^{(s)}$. Thus, we have $p \in \mathcal{V}_{D-1}$. Hence, from (5.12), $m_{D-1} = \min_{i \in \mathcal{V}_{D-1}} \hat{d}_i^{(s)} \leq \hat{d}_p^{(s)} = \hat{d}_j^{(s)} - 1 = m_D - 1$, as desired.

Next, we show that $\forall \ell \in \{1, 2, \ldots, D-1\}$, $m_\ell \leq m_{\ell+1} - 1$ implies $m_{\ell-1} \leq m_\ell - 1$. Suppose $m_\ell \leq m_{\ell+1} - 1$. Let $j \in \arg\min_{i \in \mathcal{V}_\ell} \hat{d}_i^{(s)}$, so that $j \in \mathcal{V}_\ell$ and $\hat{d}_j^{(s)} = m_\ell$. Since $\ell \neq 0$ and $\mathcal{V}_0 = \{s\}$, we have $j \neq s$. By Lemma 7(c), there exists $p \in \mathcal{N}_j$ such that $\hat{d}_p^{(s)} = \hat{d}_j^{(s)} - 1$. Note that $p \notin \mathcal{V}_\ell$, because if $p \in \mathcal{V}_\ell$, then the statement $\hat{d}_p^{(s)} = \hat{d}_j^{(s)} - 1$ contradicts the statement $j \in \arg\min_{i \in \mathcal{V}_\ell} \hat{d}_i^{(s)}$. Hence, we have either $p \in \mathcal{V}_{\ell+1}$ or $p \in \mathcal{V}_{\ell-1}$. Note that $p \notin \mathcal{V}_{\ell+1}$, because if $p \in \mathcal{V}_{\ell+1}$, then from (5.12), we have $m_{\ell+1} = \min_{i \in \mathcal{V}_{\ell+1}} \hat{d}_i^{(s)} \leq \hat{d}_p^{(s)} = \hat{d}_j^{(s)} - 1 = m_\ell - 1$, which contradicts our hypothesis that $m_\ell \leq m_{\ell+1} - 1$. Therefore, we have $p \in \mathcal{V}_{\ell-1}$. Hence, $m_{\ell-1} = \min_{i \in \mathcal{V}_{\ell-1}} \hat{d}_i^{(s)} \leq \hat{d}_p^{(s)} = m_\ell - 1$, as desired. □

Lemma 11. *For each $\ell \in \{0, 1, \ldots, D\}$, $m_\ell \geq \ell$.*

Proof. By induction. From Lemma 8, $m_0 = 0$. Next, we show that $\forall \ell \in \{1, 2, \ldots, D\}$, $m_{\ell-1} \geq \ell - 1$ implies $m_\ell \geq \ell$. Suppose $m_{\ell-1} \geq \ell - 1$. Then, by Lemma 10, $m_\ell \geq m_{\ell-1} + 1 \geq \ell$, as desired. □

Applying (5.12), (5.13), and Lemmas 9 and 11, we have $m_\ell = M_\ell = \ell \ \forall \ell \in \{0, 1, \ldots, D\}$. It follows that $\hat{d}_i^{(s)} = d_i^{(s)} \ \forall i \in \mathcal{V}$. This, together with (4.8), (4.11), (5.6), (5.9), and Lemma 7(a), implies that $\hat{x}_{(i,j)}^{(s)} = x_{(i,j)}^{(s)} \ \forall (i,j) \in \tilde{\mathcal{E}}$. This completes the proof of Theorem 12.

5.5 Distributed Estimation of Closeness

In this section, we leverage the results from Section 5.4 to derive a distributed algorithm for estimating closeness.

As is stated in Theorem 12, for any fixed node $s \in \mathcal{V}$, constraints (5.6)–(5.10) admit a unique feasible point in the mixed-integer space $\{0,1\}^{2L} \times \mathbb{R}^N$, which is exactly

the $2L + N$ variables $x_{(i,j)}^{(s)}$ $\forall (i,j) \in \tilde{\mathcal{E}}$ and $d_i^{(s)}$ $\forall i \in \mathcal{V}$. Thus, a straightforward way to develop a distributed algorithm for computing the C_i's is to make use of (1.2) and constraints (5.6)–(5.10). This algorithm, however, would have a memory complexity of $O(|\mathcal{N}_i|N)$ (where $|\cdot|$ denotes the cardinality of a set), which is rather high, since each node $i \in \mathcal{V}$ would have to compute $x_{(i,j)}^{(s)}$ $\forall j \in \mathcal{N}_i$ $\forall s \in \mathcal{V}$ and $d_i^{(s)}$ $\forall s \in \mathcal{V}$.

To derive an algorithm that has a lower memory complexity, we aggregate the $(2L + N)N$ variables $x_{(i,j)}^{(s)}$ $\forall i \in \mathcal{V}$ $\forall j \in \mathcal{N}_i$ $\forall s \in \mathcal{V}$ and $d_i^{(s)}$ $\forall i \in \mathcal{V}$ $\forall s \in \mathcal{V}$ in the following manner: first, for each $(i,j) \in \tilde{\mathcal{E}}$, in (4.18), we let

$$x_{(i,j)} \triangleq \sum_{s \in \mathcal{V}} x_{(i,j)}^{(s)}$$

so that $x_{(i,j)}$ may be regarded as the number of nodes in $\mathcal{V}$ to which shortest paths from node i pass through edge $(i,j) \in \tilde{\mathcal{E}}$. Second, for each $i \in \mathcal{V}$, recall from (4.20) that $F_i \triangleq \sum_{s \in \mathcal{V}} d_i^{(s)}$. With these two aggregations, we obtain a new set of $2L + N$ variables $x_{(i,j)}$ $\forall (i,j) \in \tilde{\mathcal{E}}$ and F_i $\forall i \in \mathcal{V}$, which has a lower memory complexity of $O(|\mathcal{N}_i|)$.

The following two lemmas show that the new variables $x_{(i,j)}$'s and F_i's satisfy a number of constraints:

Lemma 12. *The $2L$ variables $x_{(i,j)}$ $\forall (i,j) \in \tilde{\mathcal{E}}$ satisfy the following $N + L$ linear inequalities:*

$$\sum_{j \in \mathcal{N}_i} x_{(i,j)} \geq N - 1, \quad \forall i \in \mathcal{V},$$

$$x_{(i,j)} + x_{(j,i)} \leq N, \quad \forall \{i,j\} \in \mathcal{E}.$$

Proof. See Theorem 9 of Chapter 4. □

Lemma 13. *The $2L+N$ variables $x_{(i,j)}$ $\forall (i,j) \in \tilde{\mathcal{E}}$ and F_i $\forall i \in \mathcal{V}$ satisfy the following*

L linear equations:

$$F_i - F_j = x_{(i,j)} - x_{(j,i)}, \quad \forall \{i,j\} \in \mathcal{E}.$$

Proof. See Theorem 9 of Chapter 4. □

Although aggregating the variables helps lower the memory complexity, it also yields a notable drawback, in that the constraints in Lemmas 12 and 13 do not uniquely determine the new variables $x_{(i,j)}$'s and F_i's. In other words, the feasible set defined by contains in Lemmas 12 and 13 more than one point, so that Property 4 of Section 5.3 does not hold. To alleviate this drawback, consider the following lemma, which provides additional constraints that reduce the size of the feasible set:

Lemma 14. *The* $2L$ *variables* $x_{(i,j)}$ $\forall (i,j) \in \tilde{\mathcal{E}}$ *satisfy the following* $6L$ *linear inequalities:*

$$\begin{aligned} x_{(i,j)} &\geq |\mathcal{N}_j / \mathcal{N}_i|, \\ x_{(i,j)} &\leq N - |\mathcal{N}_i|, \\ x_{(i,j)} &\leq 1 + \sum_{k \in \mathcal{N}_j / (\mathcal{N}_i \cup \{i\})} x_{(j,k)}. \end{aligned}$$

Proof. See Theorem 9 of Chapter 4. □

With the introduction of these additional constraints in Lemma 14, we now have $2L + N$ variables $x_{(i,j)}$ $\forall (i,j) \in \tilde{\mathcal{E}}$ and F_i $\forall i \in \mathcal{V}$ with $8L + N$ linear equalities/ inequalities, which are an improvement over Lemmas 12 and 13 alone but are still insufficient. To alleviate this limitation, we formulate a distributed optimization problem in the following manner: first, suppose each node $i \in \mathcal{V}$ maintains estimates $\hat{x}_{(i,j)}(t) \in \mathbb{R}$ $\forall j \in \mathcal{N}_i$ and $\hat{F}_i(t) \in \mathbb{R}$ in its local memory for a memory complexity of $O(|\mathcal{N}_i|)$, where $\hat{F}_i(t)$ represents node i's estimate of its own farness from which it can

estimate its own closeness. Next, consider an objective function $f(\hat{\mathbf{x}}, \hat{\mathbf{F}})$, defined as

$$\begin{aligned} f(\hat{\mathbf{x}}, \hat{\mathbf{F}}) &= \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{N}_i} \hat{x}_{(i,j)} + \frac{1}{2}\rho \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{N}_i} \hat{x}_{(i,j)}^2 \\ &+ \frac{1}{2}\rho \sum_{i\in\mathcal{V}} \hat{F}_i^2 + \gamma_1 \sum_{i\in\mathcal{V}} \varphi\Big(N-1-\sum_{j\in\mathcal{N}_i} \hat{x}_{(i,j)}\Big) \\ &+ \gamma_2 \sum_{\{i,j\}\in\mathcal{E}} \varphi(\hat{x}_{(i,j)} + \hat{x}_{(j,i)} - N) \\ &+ \frac{1}{2}\gamma_3 \sum_{\{i,j\}\in\mathcal{E}} (\hat{F}_i - \hat{F}_j - \hat{x}_{(i,j)} + \hat{x}_{(j,i)})^2 \\ &+ \gamma_4 \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{N}_i} \varphi(|\mathcal{N}_j/\mathcal{N}_i| - \hat{x}_{(i,j)}) \\ &+ \gamma_5 \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{N}_i} \varphi(\hat{x}_{(i,j)} - N + |\mathcal{N}_i|) \\ &+ \gamma_6 \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{N}_i} \varphi\Big(\hat{x}_{(i,j)} - 1 - \sum_{k\in\mathcal{N}_j/(\mathcal{N}_i\cup\{i\})} \hat{x}_{(j,k)}\Big), \end{aligned} \tag{5.14}$$

where $\hat{\mathbf{x}} \in \mathbb{R}^{2L}$ is a vector obtained by stacking the $2L$ estimates $\hat{x}_{(i,j)}$'s, $\hat{\mathbf{F}} \in \mathbb{R}^N$ is a vector obtained by stacking the N estimates $\hat{F}_i$'s, $\rho > 0$ is a small weighting factor intended for regularization, $\gamma_1, \gamma_2, \ldots, \gamma_6 > 0$ are large weighting factors, and $\varphi(z)$ is a function defined as

$$\varphi(z) = \begin{cases} \frac{1}{2}z^2, & z \geq 0, \\ 0, & z < 0. \end{cases} \tag{5.15}$$

The objective function (5.14) has the following interpretation. The first term in (5.14) is inspired by the observation that shortest paths in graphs tend to yield small $x_{(i,j)}$'s. The second and third terms are quadratic regularization terms intended to ensure that the optimization problem is strongly convex and thus has a unique solution. The rest of the terms in (5.14) are inspired by the barrier method of converting a constrained optimization problem into an unconstrained one (thus, the γ_i's should

be large). Thus, the problem (5.14) is a regularized linear program that can be solved over the network by using, say, a gradient method, because all the terms in the objective function are distributed in nature. Taking the gradient of the objective function (5.14) with respect to the optimization variables $\hat{\mathbf{x}}$ and $\hat{\mathbf{F}}$, we obtain a scalable continuous-time distributed algorithm that operates as follows: For each $i \in \mathcal{V}$ and $j \in \mathcal{N}_i$,

$$\begin{aligned}
\dot{\hat{x}}_{(i,j)}(t) = -\varepsilon \Big[& 1 + \rho \hat{x}_{(i,j)}(t) - \gamma_1 \varphi' \Big(N - 1 - \sum_{k \in \mathcal{N}_i} \hat{x}_{(i,k)}(t) \Big) \\
& + \gamma_2 \varphi'(\hat{x}_{(i,j)}(t) + \hat{x}_{(j,i)}(t) - N) \\
& - \gamma_3 (\hat{F}_i(t) - \hat{F}_j(t) - \hat{x}_{(i,j)}(t) + \hat{x}_{(j,i)}(t)) \\
& - \gamma_4 \varphi'(|\mathcal{N}_j / \mathcal{N}_i| - \hat{x}_{(i,j)}(t)) \\
& + \gamma_5 \varphi'(\hat{x}_{(i,j)}(t) - N + |\mathcal{N}_i|) \\
& + \gamma_6 \varphi' \Big(\hat{x}_{(i,j)}(t) - 1 - \sum_{k \in \mathcal{N}_j / (\mathcal{N}_i \cup \{i\})} \hat{x}_{(j,k)}(t) \Big) \\
& - \gamma_6 \sum_{h \in \mathcal{N}_i / (\mathcal{N}_j \cup \{j\})} \varphi' \Big(\hat{x}_{(h,i)}(t) - 1 - \sum_{k \in \mathcal{N}_i / (\mathcal{N}_h \cup \{h\})} \hat{x}_{(i,k)}(t) \Big) \Big],
\end{aligned} \tag{5.16}$$

where

$$\varphi'(z) = \begin{cases} z, & z \geq 0, \\ 0, & z < 0, \end{cases} \tag{5.17}$$

and $\varepsilon > 0$ is a small positive weighting factor. For each $i \in \mathcal{V}$, we have

$$\begin{aligned}
\dot{\hat{F}}_i(t) = -\varepsilon \Big[& \rho \hat{F}_i(t) + \gamma_3 \sum_{j \in \mathcal{N}_i} (\hat{F}_i(t) - \hat{F}_j(t) \\
& - \hat{x}_{(i,j)}(t) + \hat{x}_{(j,i)}(t)) \Big],
\end{aligned} \tag{5.18}$$

$$\hat{C}_i(t) = (N-1)/\hat{F}_i(t). \tag{5.19}$$

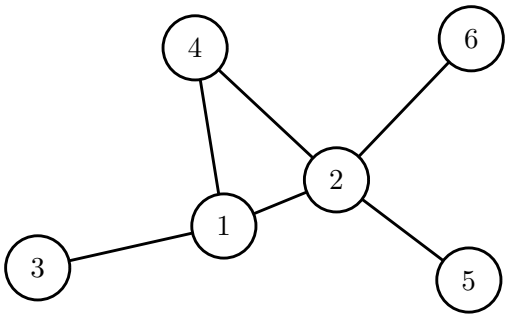
Because the problem is strongly convex, the algorithm is guaranteed to converge to the global optimizer. Indeed, Figure 5.1 illustrates the performance of algorithm (5.16)–(5.19) in estimating closeness on a 6-node graph, where it can be seen that the estimates $\hat{x}_{(i,j)}(t)$'s and $\hat{F}_i(t)$'s converge asymptotically to their true values $x_{(i,j)}$'s and F_i's.

5.6 Performance Evaluation

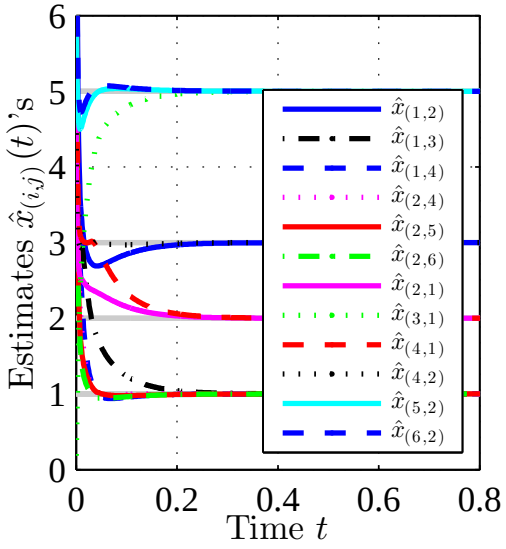
In this section, we evaluate the performance of algorithm (5.16)–(5.19) via simulation.

5.6.1 Evaluation Settings

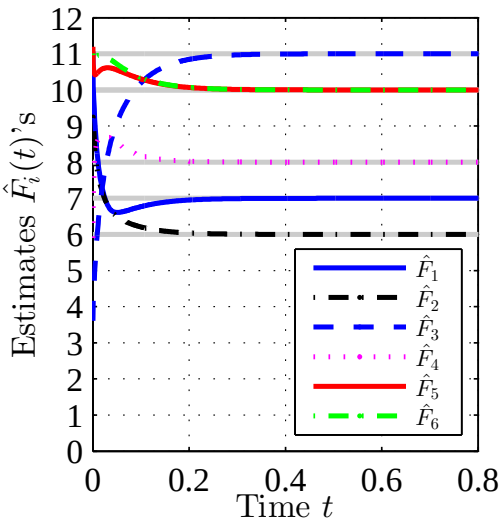
To evaluate the performance of algorithm (5.16)–(5.19), we consider three common types of random graphs, namely, *random geometric*, *Erdős-Rényi*, and *Barabási-Albert graphs* [3]. For random geometric graphs, we consider five values of N from the set $\{10, 20, 30, 40, 50\}$ and five values of neighborhood radius r from the set $\{0.1, 0.2, 0.3, 0.4, 0.5\}$. For each of the 25 combinations of (N, r), we generate 10 different scenarios. For each scenario k, we place N nodes randomly and equiprobably on the unit square $[0, 1] \times [0, 1]$, regard any two nodes as neighbors if their distance does not exceed r, and repeat the process until the resulting graph is connected. We then simulate algorithm (5.16)–(5.19) with $\varepsilon = 5 \times 10^{-5}$, $\rho = 0.001$, and $\gamma_\ell = 10^6$ $\forall \ell \in \{1, 2, \ldots, 6\}$ and record the resulting $\hat{x}_{(i,j)}(T)$ $\forall (i, j) \in \tilde{\mathcal{E}}$, $\hat{F}_i(T)$ $\forall i \in \mathcal{V}$, and $\hat{C}_i(T)$ $\forall i \in \mathcal{V}$, where T is the simulation duration. Upon completion, we compare the closeness estimates $\hat{C}_1(T), \hat{C}_2(T), \ldots, \hat{C}_N(T)$ with their true values $C_1, C_2, \ldots, C_N$ computed from (1.2). To facilitate the comparison, we introduce a measure called *Accuracy of Closeness Ordering*, denoted as $\mathrm{ACO}(N, r, k)$ (i.e., for



(a) Node indices.



(b) Estimates $\hat{x}_{(i,j)}(t)$'s.



(c) Estimates $\hat{F}_i(t)$'s.

Figure 5.1: An illustration of the performance of algorithm (5.16)–(5.19) in estimating closeness on a 6-node graph.

scenario k of combination (N, r)) and defined as

$$\mathrm{ACO}(N, r, k) = \frac{2}{N(N-1)} \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}, j > i} \mathcal{O}(\hat{C}_i(T), \hat{C}_j(T), C_i, C_j),$$

where $\mathcal{O}(a_1, a_2, b_1, b_2)$ is a binary-valued function that returns 1 if $a_1 \leq a_2$ and $b_1 \leq b_2$ or if $a_1 \geq a_2$ and $b_1 \geq b_2$ (i.e., (a_1, a_2) and (b_1, b_2) have the same order), and returns 0 otherwise (i.e., different order). Thus, the Accuracy of Closeness Ordering measures how well the closeness estimates match up with their true values in terms of ordering, takes the maximum value of 1 when the order is completely preserved, and takes the minimum value of 0 when the order is completely reversed. Upon calculating $\mathrm{ACO}(N, r, k)$ for each scenario $k \in \{1, 2, \ldots, 10\}$, we record their average $\frac{1}{10} \sum_{k=1}^{10} \mathrm{ACO}(N, r, k)$ as $\mathrm{ACO}(N, r)$.

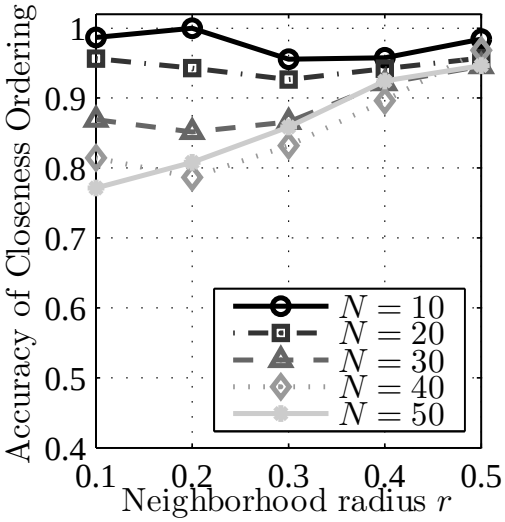
Similar to the aforementioned random geometric graphs, for Erdős-Rényi graphs we consider the same five values of N and five values of edge probability p from the set $\{0.1, 0.2, 0.3, 0.4, 0.5\}$. For each of the 25 combinations of (N, p), we generate 10 different scenarios, and for each scenario k, we let there be N nodes, let every pair of them have probability p of being neighbors, and repeat the process until the resulting graph is connected. Following the same procedure, we then simulate algorithm (5.16)–(5.19) and record the average of the resulting Accuracy of Closeness Ordering as $\mathrm{ACO}(N, p)$. Finally, for Barabási-Albert graphs, we consider the same five values of N and five values of m from the set $\{1, 2, 3, 4, 5\}$, which represents the number of edges a new node has when it joins an existing graph. As before, for each combination of (N, m), we generate 10 different scenarios, and for each scenario k, we begin with an initial graph of $m + 1$ nodes and carry out the preferential attachment process [3] by repeatedly attaching a new node to m of the existing nodes until the graph has N nodes. Again following the same procedure, we record the average of the resulting Accuracy of Closeness Ordering as $\mathrm{ACO}(N, m)$.

5.6.2 Evaluation Results

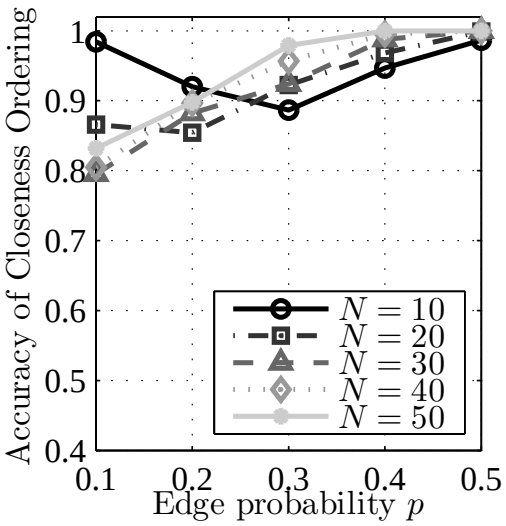
Figure 5.2 displays the evaluation results. Specifically, Figure 5.2(a) represents the Accuracy of Closeness Ordering ACO(N, r) as a function of the number of nodes N and the neighborhood radius r for random geometric graphs. Likewise, Figure 5.2(b) plots ACO(N, p) as a function of N and the edge probability p for Erdős-Rényi graphs, while Figure 5.2(c) plots ACO(N, m) as a function of N and the number of new edges m for Barabási-Albert graphs. Analyzing these figures, we see that the proposed distributed algorithm performs well, achieving an Accuracy of Closeness Ordering score of 0.91 on average and 0.76 at worst, in all the scenarios considered. This means that despite knowing only who their neighbors are, with algorithm (5.16)–(5.19) the nodes are able to estimate their closeness with a 91% accuracy in terms of ordering. As another observation, we see that the performance of the algorithm is best when the graph is either sufficiently dense, or extremely sparse.

5.7 Conclusion

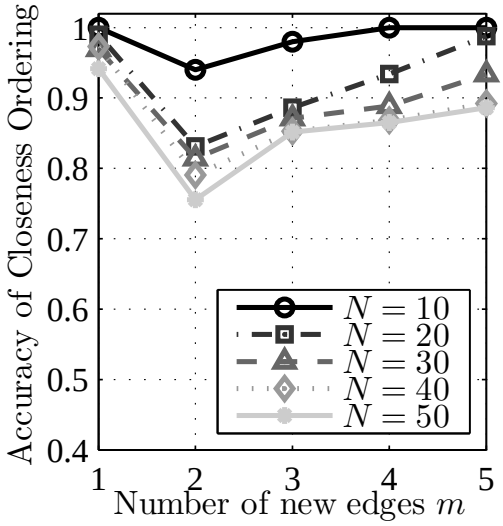
In this chapter, we have developed a scalable distributed algorithm that enables nodes in a general graph to cooperatively estimate their individual closeness with only local interaction and without any centralized coordination, nor high memory usages. We have also shown via extensive simulation on three common random graphs that this algorithm is fairly accurate, but has room for improvement. Therefore, possible future research directions include deriving additional constraints that decrease the size of the feasible set and incorporating them into the algorithm to increase its estimation accuracy.



(a) Random geometric graphs.



(b) Erdős-Rényi graphs.



(c) Barabási-Albert graphs.

Figure 5.2: Performance of algorithm (5.16)–(5.19) on three common types of random graphs as measured by the Accuracy of Closeness Ordering.

Chapter 6

Applications of Betweenness Centrality on Community Detection and Information Spreading

In the previous chapters, we developed a collection of scalable distributed algorithms, which enable nodes in a large-scale network to cooperatively learn how important they are individually, with only local interaction and with neither global coordination nor knowledge of the network topology using the most fundamental centrality measures from the area of complex networks, namely, the betweenness centrality and closeness centrality, as well as a subset of their variations.

In this chapter, we experimented our algorithms for estimating node and edge betweenness centralities on both computer generated graphs and real networks for community detection and information spreading, and then proposed a method for spreading information with the knowledge of community structures. These real network data sets are maintained by Prof. Tim Davis of Texas A&M University and

Dr. Yifan Hu of Yahoo Labs. The data sets are available at http://www.cise.ufl.edu/research/sparse/matrices [111].

The outline of this chapter is as follows: Section 6.1 implements a clustering algorithm for community detection using both the edge betweenness and the estimated edge betweenness which is defined and calculated using algorithm proposed in Chapter 4. We then evaluate the edge betweenness based and estimated edge betweenness based algorithms' performance via simulation on both computer generated and real networks. Section 6.2 implements a simple information spreading algorithm using the ranking of node betweenness, estimated node betweenness and closeness on several computer generated and real networks, with a goal of evaluating different centrality measures for the identification of influential people to spread information. The estimated node betweenness is defined and calculated using algorithm proposed in Chapter 4. Furthermore, based on the algorithms introduced in Sections 6.1 and 6.2, Section 6.3 proposes a combined algorithm to identify a certain number of influential people for efficient spreading information, and evaluates the performance via simulation using betweenness centrality and estimated betweenness on both computer generated and real networks. Finally, Section 6.4 concludes this chapter.

6.1 Detecting Community

A social network is a graph of relationships between individuals, groups, organizations, or even entire societies. There are a wide variety of such groups or organizations in our social network, such as families, friends, and collaborators, just to name a few. To understand the community structures of a social network and determine how people interact and form groups, Girvan and Newman proposed an algorithm [122] in a seminal paper appeared in 2002. Communities are detected in a network after some iterations of identifying and removing the edges lying between them. The edges be-

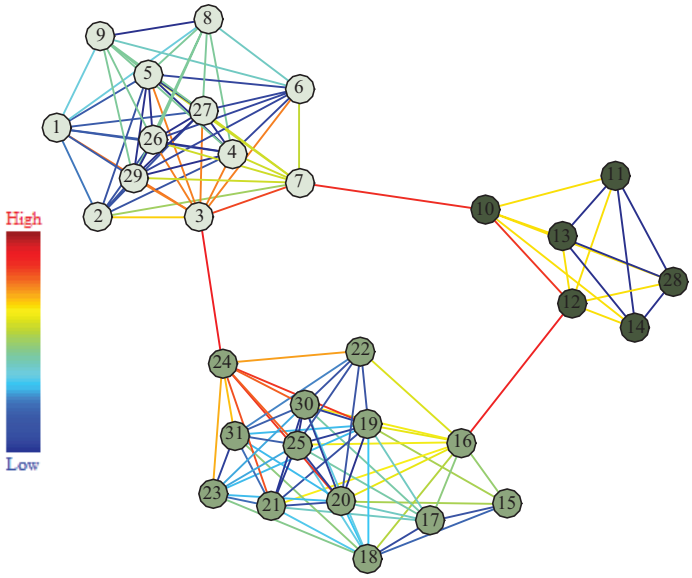


Figure 6.1: An illustration of edge betweenness in community detection

tween communities are detected according to the values of the edge betweenness. The key idea of the algorithm is that a few edges that lie between highly clustered communities can be viewed as bottlenecks between the communities. Thus all the communications between members of different communities would very likely go through these edges, resulting in the few edges with high edge betweenness values. Figure 6.1 shows an illustration of edge betweenness in a network. The number shown in each node is the node index, and the color of each edge represents how important an edge is in terms of its edge betweenness, where red represents high value and blue represents low value, as shown in the color bar. The three different colors indicate nodes belonging to different communities. Notice that cutting these bottlenecks edges {7, 10}, {3, 24}, and {12, 16}, which have the highest edge betweenness, would result in three separate communities in this network.

The Girvan-Newman algorithm [122] proceeds as follows:

- Repeat until no edges are left:
  - Calculate betweenness for all edges in the network.
  - Identify and remove the edge with the highest betweenness value. If all

the edges within a group or the entire network have the same betweenness value, randomly select and remove one of these edges.

- Connected components are communities.

To test the performance of an algorithm and to identify good partitions, we compute the modularity, the most popular quality function defined by Newman and Girvan in [123], as

$$Q = \sum_{c_i=1}^{n} (e_{c_i c_i} - a_{c_i}^2), \tag{6.1}$$

where n is the number of communities; $e_{c_i c_j}$ represents the fraction of all edges in the network that connect nodes in community c_i to nodes in community c_j; and $a_{c_i} = \sum_{c_j=1}^{n} e_{c_i c_j}$ represents the fraction of edges that connect to nodes in community c_i. Thus, $\sum_{c_i=1}^{n} e_{c_i c_i}$ gives the fraction of edges in the network that connect nodes in the same community, and clearly a good division into communities should have a high value of $\sum_{c_i=1}^{n} e_{c_i c_i}$. This quantity measures the fraction of the edges in the network that connect nodes in the same community (i.e., internal edges) minus the expected value of the same quantity in a network with the same community divisions but random connections between the nodes [123]. If the number of internal edges is larger than the random value, we will get $Q > 0$. Values approaching $Q = 1$, which is the maximum possible value, indicate a strong community structure.

Having an assumption that a random graph is not expected to have a community structure, (6.1) indicates that the possible existence of communities could be revealed by the comparison between the actual density of edges in a subgraph and the density one would expect to have in the subgraph if the nodes of the graph were attached regardless of community structure. This expected edge density depends on the original graph keeping some of its structural properties but without community structure.

Modularity can then be rewritten in [124] as follows

$$Q = \sum_{c=1}^{n} [\frac{l_c}{m} - (\frac{d_c}{2m})^2], \tag{6.2}$$

where m is the number of edges in the original graph; n is the number of communities; l_c is the total number of edges joining nodes of community c; and d_c is the sum of the degrees of the nodes in c. In (6.2), the first term of each summand is the fraction of internal edges of the community, whereas the second term represents the expected fraction of edges that would be there if the graph were a random graph with the same expected degree for each node.

As indicated in [124], a subgraph is a community if the corresponding contribution to modularity in the sum is positive. The more the number of internal edges of the community exceeds the expected number, the better the community is defined. So, large positive values of the modularity indicate good partitions. Notice that the modularity of the whole graph, taken as a single community, is zero, as the two terms in (6.2) in this case are equal. Also notice that the modularity is always smaller than one, and can be negative as well. For instance, the partition in which each node is a community is always negative: in this case, the sum runs over n terms, which are all negative as the first term of each summand is zero. This is a nice feature of the measure, implying that, if there are no partitions with positive modularity, the graph has no community structure. In contrary, the existence of partitions with large negative modularity values may indicate the existence of subgroups with very few internal edges and many edges lying between them [125]. Note that the maximum modularity of a graph generally grows if the size of the graph or the number of (well-separated) communities increases [126]. Therefore, modularity should not be used to compare the quality of the community structure of graphs which are very different in size.

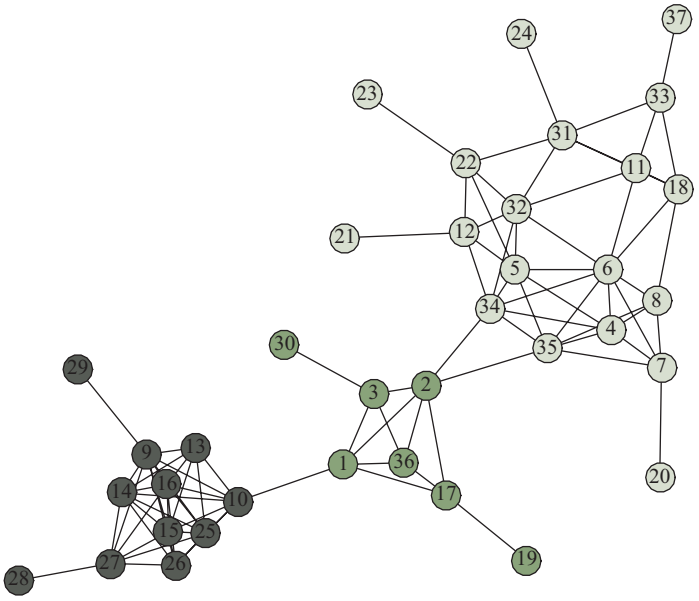
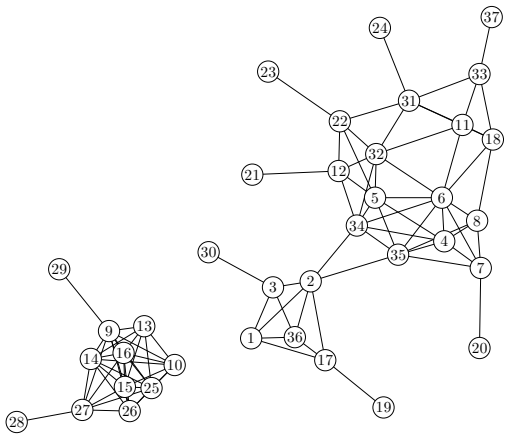


Figure 6.2: A 37-node graph with its node indices.

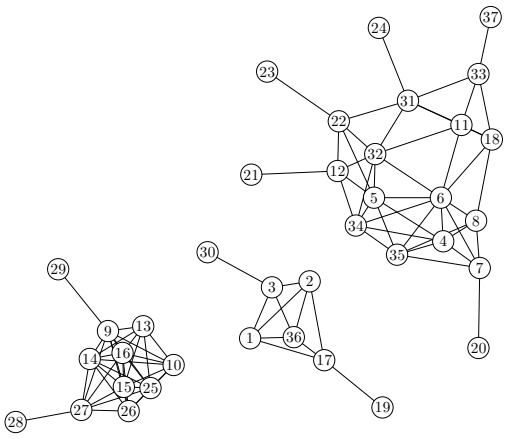
In this section, we first implement the Girvan-Newman algorithm using both the original definition of edge betweenness defined in (2.1) and the estimated edge betweenness which is defined in (4.2) and calculated using our algorithm proposed in Chapter 4 to detect communities on computer generated and real world networks. We then compute the modularity value for each network to evaluate and compare the performance of edge betweenness based and estimated betweenness based Girvan-Newman algorithms.

First, we test the algorithms on a computer generated graph, shown in Figure 6.2, which is constructed with 37 nodes and 92 edges. The number in each node represents the node index. Notice that in this network there are three communities of densely connected nodes, with a much lower density of connections between them. The three different colors indicate nodes in different communities.

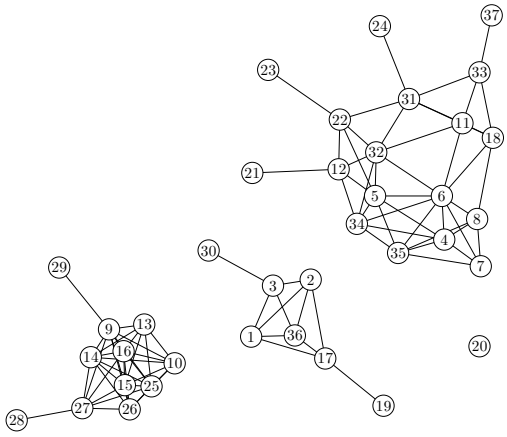
We get the same result when applying the Girvan-Newman algorithm on this network using edge betweenness and estimated edge betweenness. In fact, all the nodes are classified correctly with either method. Figure 6.3 shows the community structures obtained from edge betweenness and estimated edge betweenness based



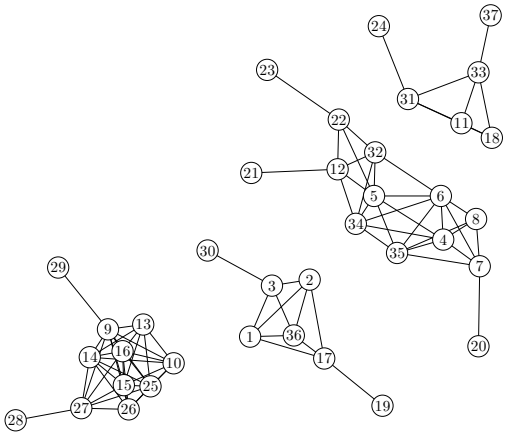
(a) $B_{\{i,j\}}$ and $\hat{B}_{\{i,j\}}$ based algorithms with $n = 2$.



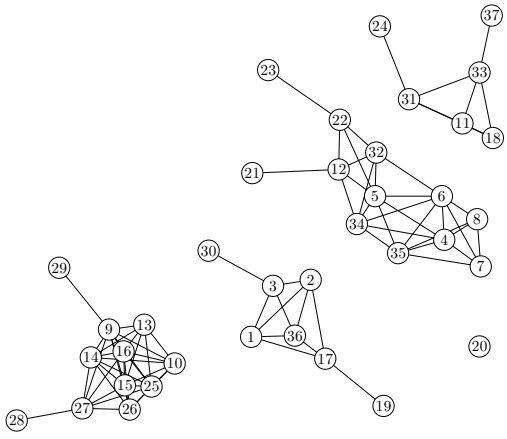
(b) $B_{\{i,j\}}$ and $\hat{B}_{\{i,j\}}$ based algorithms with $n = 3$.



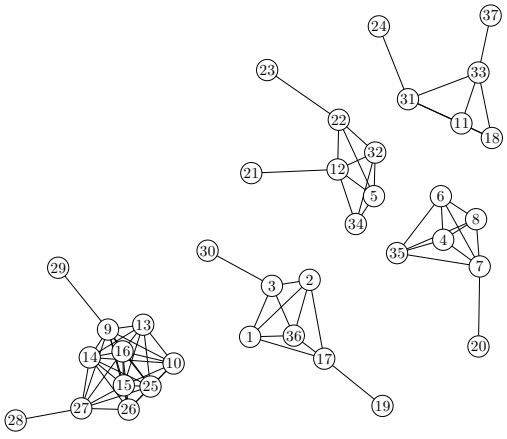
(c) $B_{\{i,j\}}$ based algorithm with $n = 4$.



(d) $\hat{B}_{\{i,j\}}$ based algorithm with $n = 4$.



(e) $B_{\{i,j\}}$ based algorithm with $n = 5$.



(f) $\hat{B}_{\{i,j\}}$ based algorithm with $n = 5$.

Figure 6.3: Community structures after edges removal on a 37-node network.

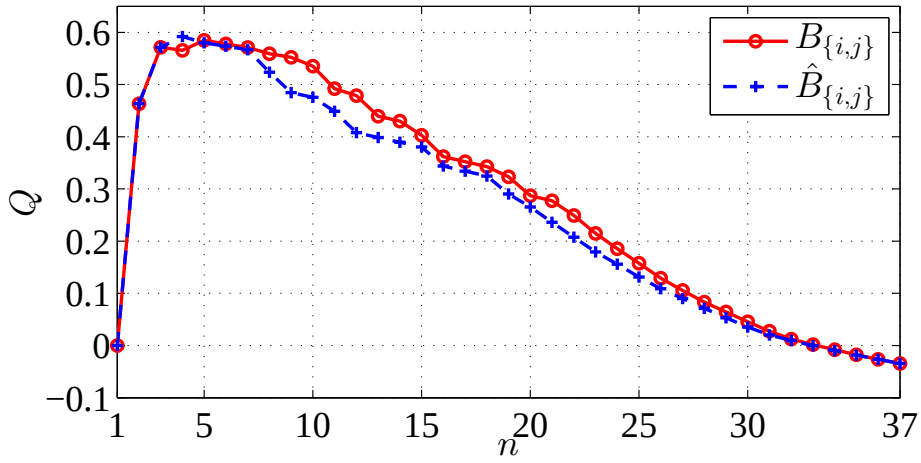


Figure 6.4: Modularity values of different clustering on a 37-node graph.

Girvan-Newman algorithms, where the number of communities n is selected from the set $\{2, 3, 4, 5\}$. Note that both algorithms achieve the same clustering result when $n = 2$ and $n = 3$, with $Q = 0.4631$ and $Q = 0.5713$, respectively, as shown in Figure 6.3(a) and (b). Figure 6.3(c) and (d) shows the community structures when $n = 4$ where (c) is obtained from the edge betweenness based algorithm with $Q = 0.5656$, and (d) is obtained from the estimated edge betweenness based algorithm with $Q = 0.5919$. In this case, the estimated edge betweenness based algorithm achieves a better clustering. Figure 6.3(e) and (f) shows the community structures when $n = 5$ where (e) is obtained from the edge betweenness based algorithm with $Q = 0.5848$, and (f) is obtained from the estimated edge betweenness based algorithm with $Q = 0.5801$. In this case, the edge betweenness based algorithm achieves a slightly better clustering.

To illustrate and compare the performance of two algorithms, Figure 6.4 shows the modularity values Q for edge betweenness based and estimated edge betweenness based Girvan-Newman algorithm, plotted as a function of the number of communities n. For edge betweenness based algorithm, the modularity Q has two maxima corresponding to splitting into three communities with $Q = 0.5713$ which agrees with our intuition, and five communities with $Q = 0.5848$, but with the fact that one of those

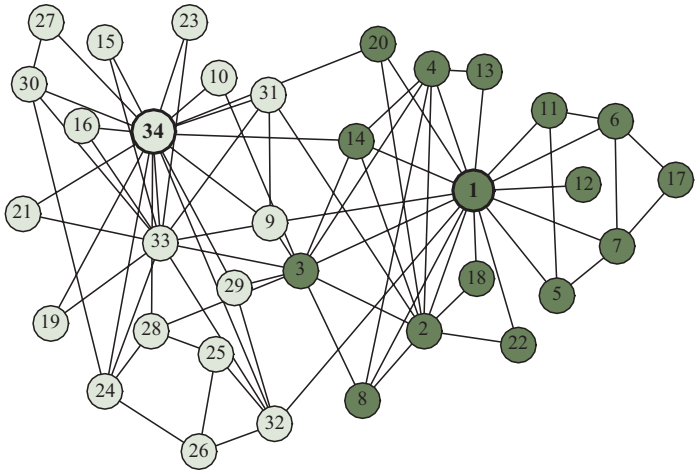


Figure 6.5: Zachary's karate club network.

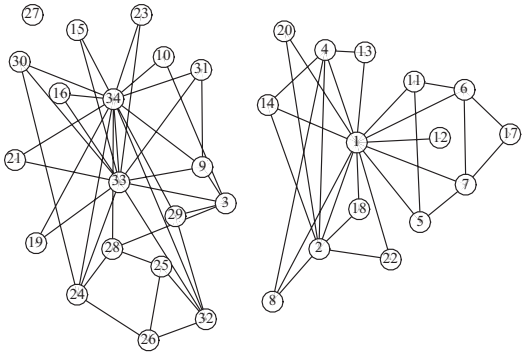
five contains only one node. For the estimated edge betweenness based algorithm, the modularity has a single peak with $Q = 0.5919$ at the point where the network breaks into four communities. Notice that when the network is split into three and five communities, the modularity values obtained from the estimated edge betweenness based algorithm are 0.5713 and 0.5801 respectively, which are close to the values obtained from the edge betweenness based algorithm.

Although computer generated networks are reproducible and well controlled for testing and comparing the performance of algorithms, it is also desirable to test the algorithms on data from real world networks. To this end, we have utilized Zachary's karate club network [127], which is a well-known network and regularly used as a benchmark to test community detection. This dataset is recorded by Zachary about the friendship among 34 members of a karate club at a US university in the 1970s. During Zachary's three-year observation, there was a conflict between the administrator of the club and the club's instructor, which ultimately resulted in the instructor's leaving and starting a new club, taking about a half of the original club's members with him. Figure 6.5 shows the club network, which contains 34 nodes and 78 edges. The node 1 represents the instructor and node 34 represents the administrator, and the two different colors indicate two separate groups, supporting

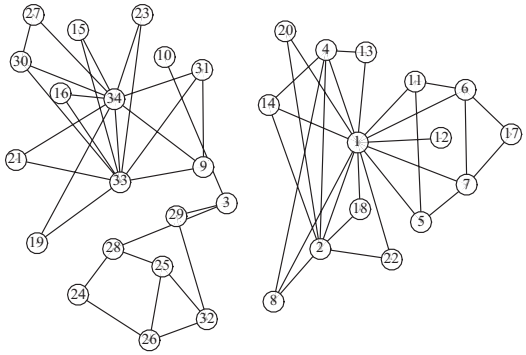
the instructor and the president, respectively.

By applying the Girvan-Newman algorithm on this network based on the edge betweenness and the estimated edge betweenness, we are able to detect the same two groups (i.e., when $n = 2$) from the original network structure, and only node 3 is classified incorrectly. Figure 6.6(a) and (b) shows the community structures when $n = 3$ where (c) is obtained from the edge betweenness based algorithm with $Q = 0.3488$, and (d) is obtained from the estimated edge betweenness based algorithm with $Q = 0.3915$. Figure 6.6(c) and (d) shows the community structures when $n = 4$ where (c) is obtained from the edge betweenness based algorithm with $Q = 0.3373$, and (d) is obtained from the estimated edge betweenness based algorithm with $Q = 0.4060$. In these two cases, the estimated edge betweenness based algorithm achieves a better clustering. Figure 6.3(e) and (f) shows the community structures when $n = 5$ where (e) is obtained from the edge betweenness based algorithm with $Q = 0.3706$, and (f) is obtained from the estimated edge betweenness based algorithm with $Q = 0.3889$. In this case, the edge betweenness based algorithm achieves a slightly better clustering. Figure 6.7 shows the modularity values for edge betweenness and the estimated edge betweenness based algorithms, plotted as a function of the number of communities n. Note that both algorithms achieve the same clustering result when $n = 2$ with $Q = 0.3600$ and misclassify only node 3. To further study the relationship among the club members, and to illustrate and compare the differences between two algorithms, Figure 6.6 shows the community structures obtained from edge betweenness and the estimated edge betweenness based Girvan-Newman algorithms, when the number of communities n is selected from the set $\{3, 4, 5\}$.

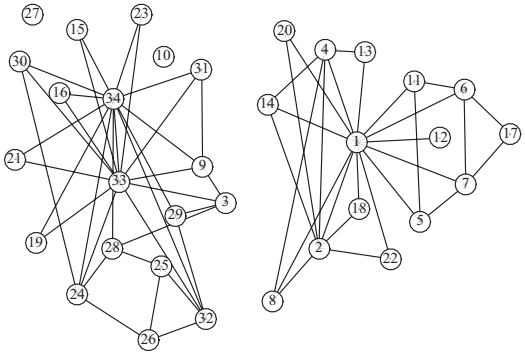
In the previous examples, we tested our algorithm on a number of networks for which the community structure was known in advance. The results indicate that both edge betweenness and estimated betweenness based algorithms are sensitive and accurate methods for extracting community structure from networks. Now, we apply



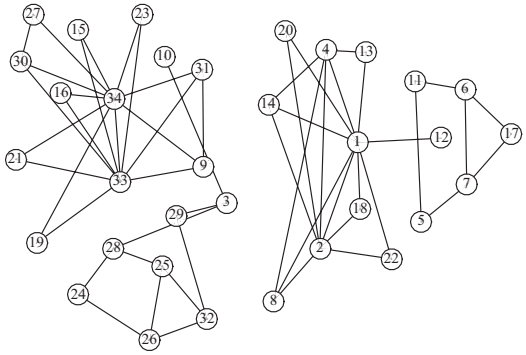
(a) $B_{\{i,j\}}$ based algorithm with $n = 3$.



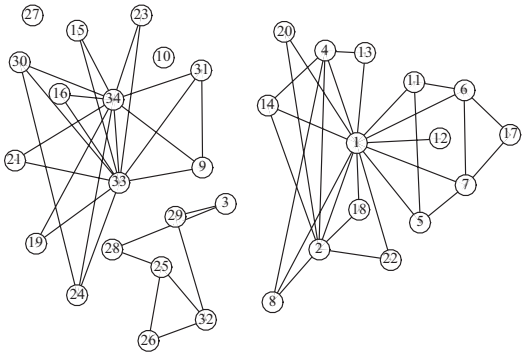
(b) $\hat{B}_{\{i,j\}}$ based algorithm with $n = 3$.



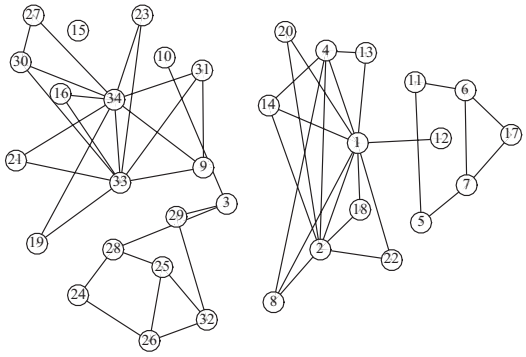
(c) $B_{\{i,j\}}$ based algorithm with $n = 4$.



(d) $\hat{B}_{\{i,j\}}$ based algorithm with $n = 4$.



(e) $B_{\{i,j\}}$ based algorithm with $n = 5$.



(f) $\hat{B}_{\{i,j\}}$ based algorithm with $n = 5$.

Figure 6.6: Community structures after edges removal on Zachary's karate club network.

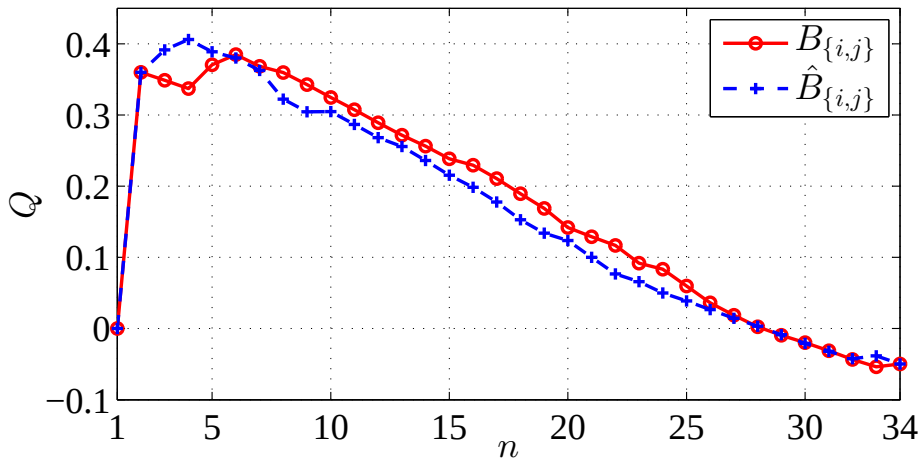


Figure 6.7: Modularity values of different clustering on Zachary's karate club network.

both algorithms to a network for which the structure is not known and show that in this case it can help us learn and understand the complex dataset. The network we have studied is a dolphin social network, which is an undirected social network of frequent associations between a community of bottlenose dolphins living off Doubtful Sound, New Zealand. This network is constructed from observations over a period of 7 years from 1994 to 2001 and compiled by Lusseau et al. [128]. Figure 6.8 shows the dolphin social network. The 62 nodes in the network represent the 62 dolphins, and 159 edges between nodes represent associations between dolphin pairs occurring more often than expected by chance. Figure 6.9 shows the modularity values for edge betweenness and the estimated edge betweenness based algorithms, plotted as a function of the number of communities n. We can see from Figure 6.9 that when $n < 10$, the estimated edge betweenness based algorithm achieves a better clustering than the edge betweenness based algorithm does, and vice versa when $n > 10$.

Based on our tests on computer generated and real word networks, and from the modularity values shown in Figures 6.4, 6.7 and 6.9, we find that the estimated edge betweenness performs very well in correctly detecting community structures in almost all scenarios, especially when the number of communities is very small compared with the size of the network.

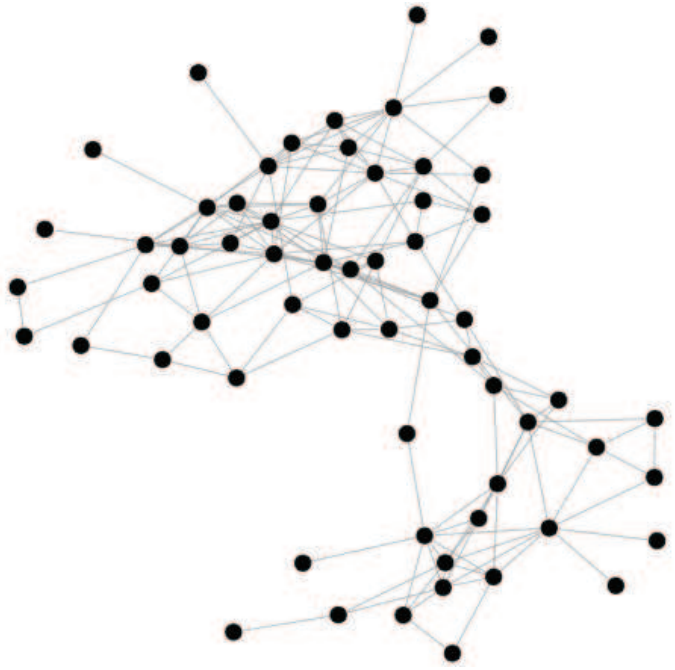


Figure 6.8: A dolphin social network (adopted from [2]).

6.2 Spreading Information

As indicated in the last section, a social network involves many different types of relations, including but not limited to communication networks, information networks, and collaboration networks. According to sociologists, a limited number of influential people in these networks are able to spread information more efficiently than other people.

Early research in this area (Leavitt [129]) indicated that the person occupying the more central position in a given network is more likely to emerge as the leader. In other words, people in more central position resulting in higher centrality scores will have more influence than people in less central positions. As remarked in [130], betweenness and closeness centrality measures imply different "theories" of how centrality might affect group processes: centrality as potential for control of communication, and as independence or efficiency. Despite this fact, both centrality measures have some features in common, e.g., they agree in assigning the highest centrality score to the

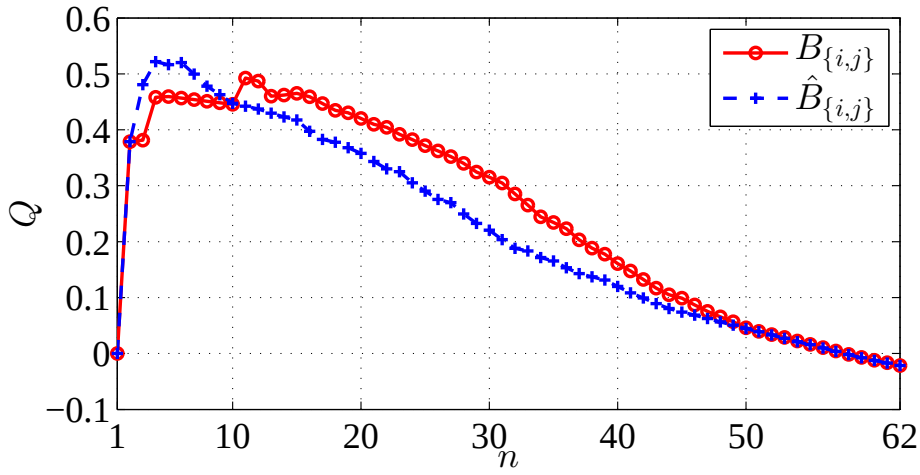
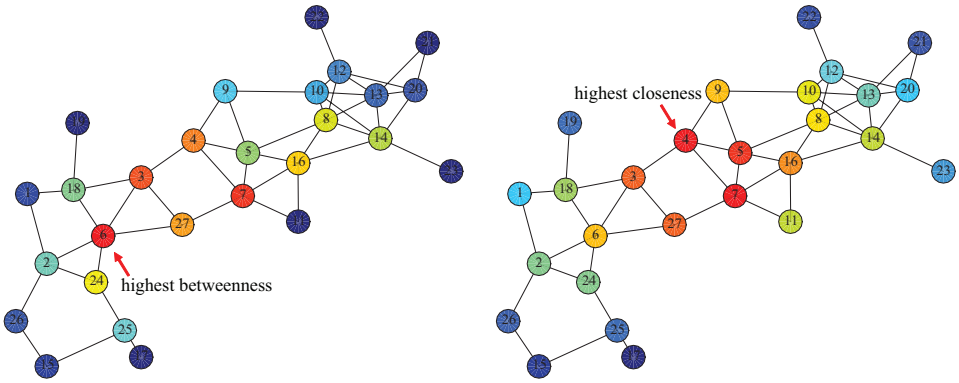


Figure 6.9: Modularity values of different clustering on the dolphin social network.



(a) Node betweenness. (b) Classic closeness.

Figure 6.10: An illustration of node betweenness and closeness.

central node in a star network, and the low centrality scores to nodes in a cycle and in complete networks. They also can be very different, e.g., a leaf node connected with the center of a network has lowest betweenness, but may have high closeness. In a given application, one centrality measure might be more appropriate than the other. An illustration of these two measures, which highlights their key attributes, is provided in Figure 6.10, where red color represents nodes with high node betweenness and classic closeness centralities, while blue color represents nodes with low values. We can see from Figure 6.10 that the importance of nodes is different for these two centrality measures in the same graph. The most important node is also different for

	high betweenness	high closeness
low betweenness		close to many other nodes in the network, but either lies in the multiple shortest paths like many other nodes do, or lies in the few shortest paths
low closeness	locates in a dominant position of communication between a large group and a small one	

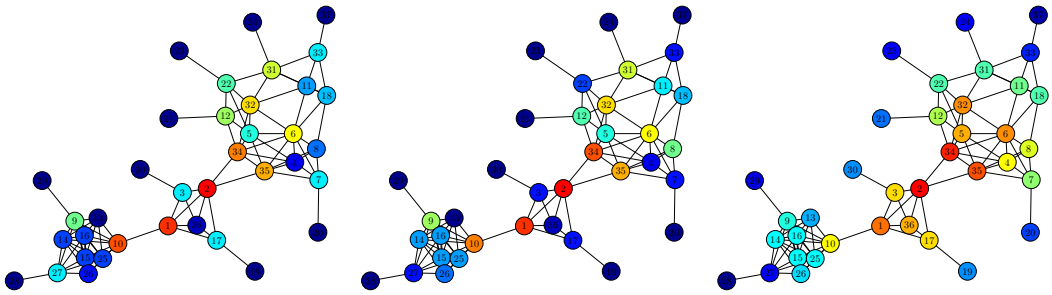
Table 6.1: Comparison of node betweenness and closeness centralities.

different measures. For example, we can see from Figure 6.10(a) that node 6 has the highest betweenness, and from Figure 6.10(b) that node 4 has the highest closeness. Table 6.1 shows a comparison of node betweenness and closeness centrality.

In this section, we compare the performance of several different centrality measures in terms of their behavior in identifying influential people for information spreading. They include the original definition of node betweenness in (1.1), estimated node betweenness in (4.1), and classic closeness in (1.2). These measures are tested on several computer generated and real networks. We assume that centrality is an appropriate measure for the identification of influential people.

To this end, we developed a simple algorithm for information spreading. The idea of the experiments is to provide information initially to n most "central" nodes in a network based on the ranking of node betweenness, estimated node betweenness and closeness values, and also to n randomly selected nodes. Then we count the number of nodes reached at a certain step k by applying a Breadth-First-Search (BFS) algorithm. We designed the experiment to show that by sending information to "central" people in a network is better than randomly selected people, in terms of information spreading. We present several experiments on both computer generated and real world networks.

First, we implement this information spreading algorithm on a 37-node computer

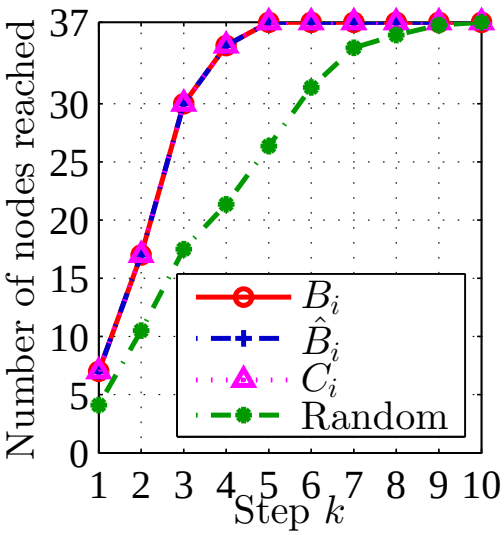


(a) B_i. (b) $\hat{B}_i$. (c) C_i.

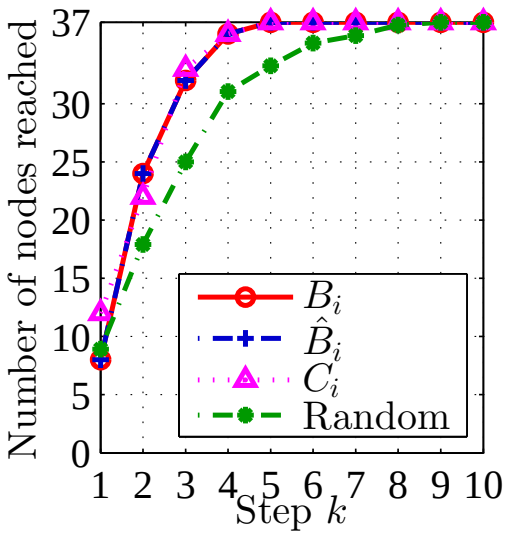
Figure 6.11: A 37-node graph with its node indices, node betweenness, estimated node betweenness and closeness centrality.

generated graph, as shown in Figure 6.11, where the numbers and colors inside each node represent their indices and importance. Importance here is represented by different centrality measures for different figures, i.e., it refers to node betweenness in Figure 6.11(a), estimated node betweenness in Figure 6.11(b), and closeness centrality in Figure 6.11(c). Figure 6.12 displays the comparison and evaluation results of algorithms using these three centrality measures and a random initial nodes selection algorithm. In this experiment, we consider four numbers of initially selected nodes from the set $\{1, 2, 3, 4\}$. Specifically, Figure 6.12(a) represents the number of nodes reached as a function of step k when only one node is selected initially to spread information. Likewise, Figure 6.12(b) plots the number of nodes reached as a function of step k when two nodes are selected initially, while Figure 6.12(c) plots the number of nodes reached as a function of step k when three nodes are selected initially. Figure 6.12(d) shows the number of nodes reached as a function of step k when four nodes are selected initially. To avoid bias in the evaluation of the random selection we applied this algorithm 20 times on the network and calculated averages. From these figures, we see that information spreads faster using centrality measures based initial nodes selection algorithms in all scenarios and the one using the estimated node betweenness achieves the best performance in these four scenarios in this network.

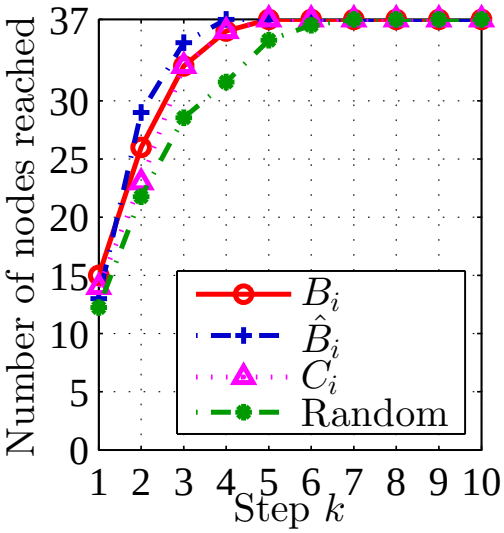
We next implement this information spreading algorithm on Zachary's karate club



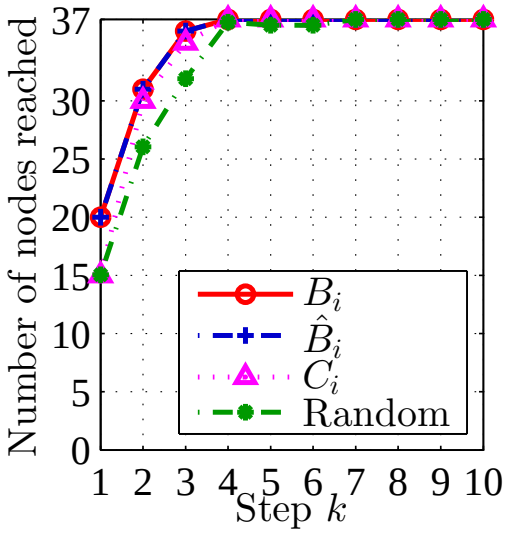
(a) $n = 1$.



(b) $n = 2$.

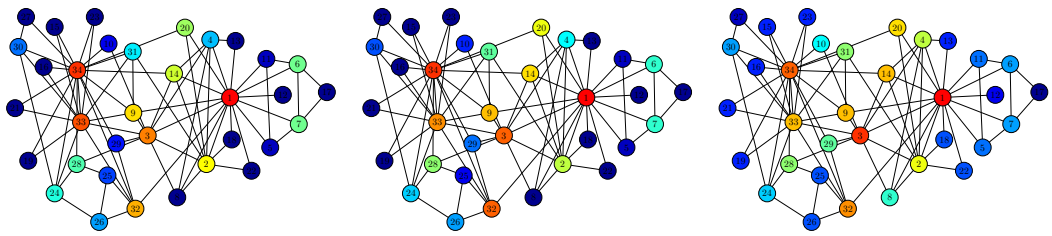


(c) $n = 3$.



(d) $n = 4$.

Figure 6.12: An information spreading algorithm on a 37-node network.

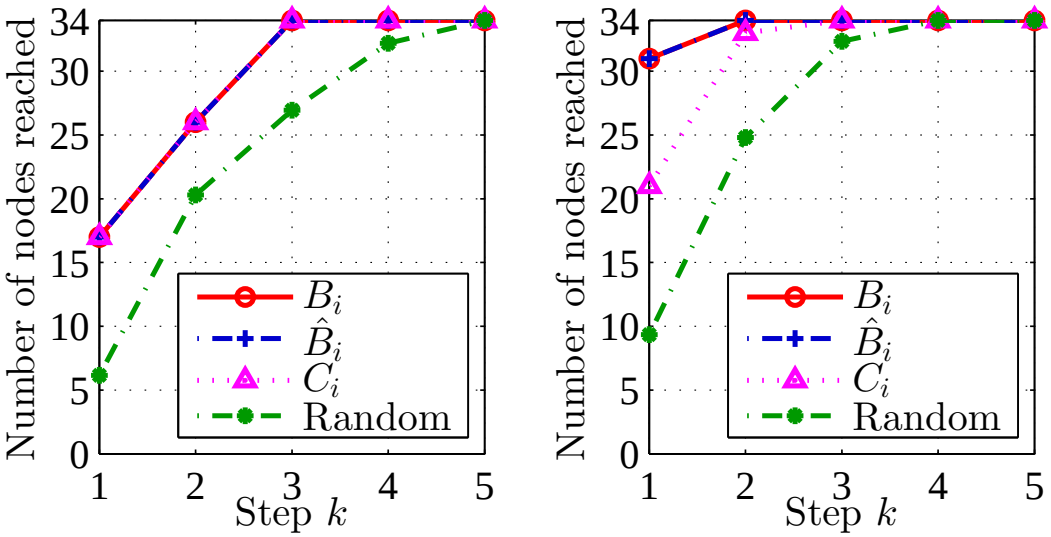


(a) B_i.

(b) $\hat{B}_i$.

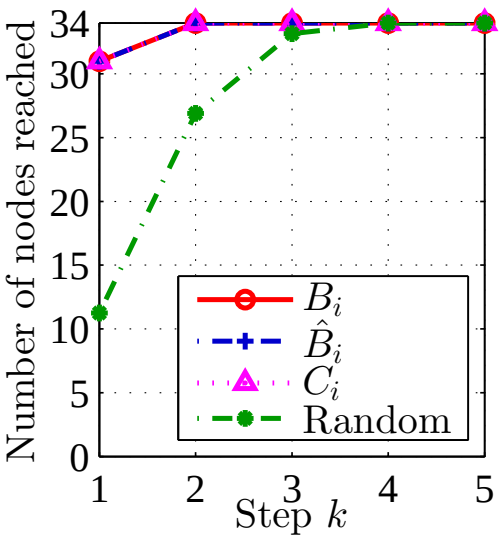
(c) C_i.

Figure 6.13: Zachary's karate club network with its node indices, node betweenness, estimated node betweenness and closeness centrality.

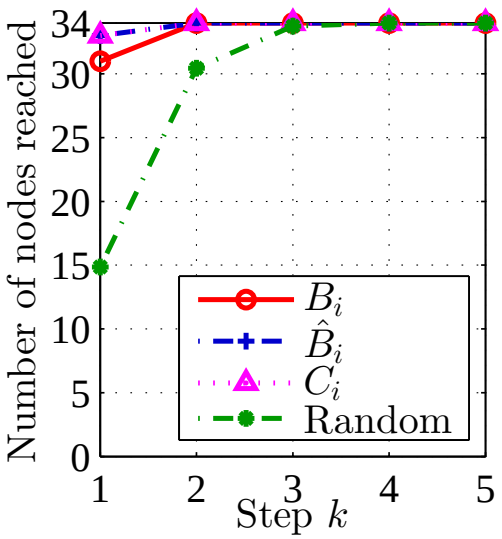


(a) $n = 1$.

(b) $n = 2$.



(c) $n = 3$.

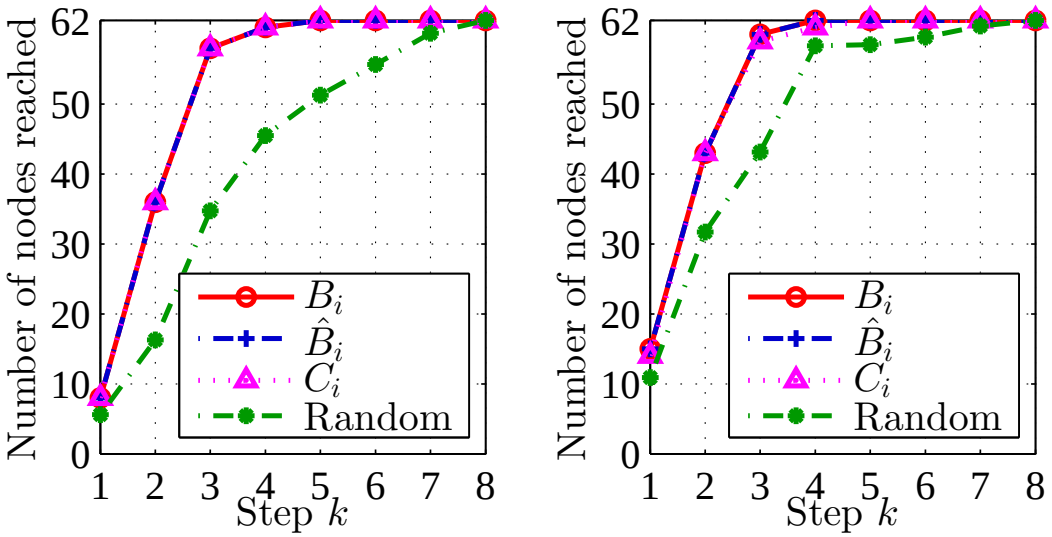


(d) $n = 4$.

Figure 6.14: An information spreading algorithm on Zachary's karate club network.

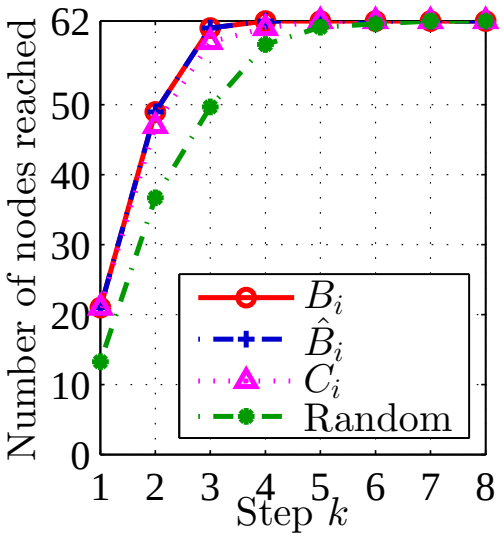
network, as shown in Figure 6.13, where the numbers and colors inside each node represent their indices and importance. Importance is represented by different centrality measures for different figures, i.e., it refers to node betweenness in Figure 6.13(a), estimated node betweenness in Figure 6.13(b), and closeness centrality in Figure 6.13(c). Figure 6.14 displays the comparison and evaluation results of algorithms using these three centrality measures and a random initial nodes selection algorithm. In this experiment, we consider four numbers of initially selected people from the set $\{1, 2, 3, 4\}$. Specifically, Figure 6.14(a) represents the number of people reached as a function of step k when only one person is selected initially to spread information. Likewise, Figure 6.14(b) plots the number of people reached as a function of step k when two people are selected initially, while Figure 6.14(c) plots the number of people reached as a function of step k when three people are selected initially. Figure 6.14(d) shows the number of people reached as a function of step k when four people are selected initially. To avoid bias in the evaluation of the random selection we applied this algorithm 20 times on the network and calculated averages. From these figures, we see that information spreads faster using centrality measures based initial people selection algorithms, and the one using estimated node betweenness achieves the best performance in these four scenarios in this network.

We then tested our algorithm on the dolphin social network. Figure 6.15 displays the comparison and evaluation results for different centrality measures based initial dolphins selection algorithms and a random initial dolphins selection algorithm. In this experiment, we consider four numbers of initially selected dolphins from the set $\{1, 2, 3, 4\}$. Specifically, Figure 6.15(a) represents the number of dolphins reached as a function of step k when only one dolphin is selected initially to spread information. Likewise, Figure 6.15(b) plots the number of dolphins reached as a function of step k when two dolphins are selected initially, while Figure 6.15(c) plots the number of dolphins reached as a function of step k when three dolphins are selected initially.

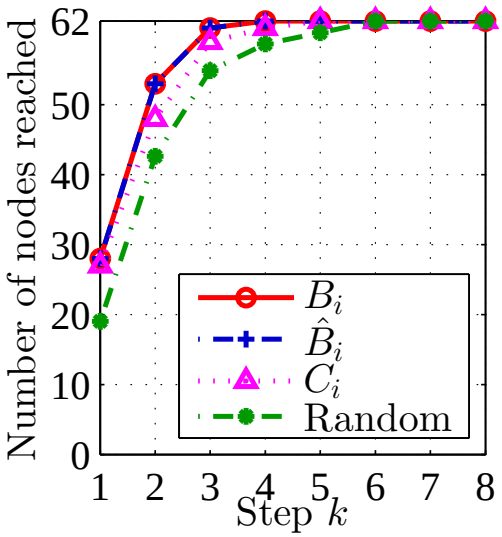


(a) $n = 1$.

(b) $n = 2$.



(c) $n = 3$.



(d) $n = 4$.

Figure 6.15: An information spreading algorithm on the dolphin social network.

Figure 6.15(d) shows the number of dolphins reached as a function of step k when four dolphins are selected initially. To avoid bias in the evaluation of the random selection we applied this algorithm 20 times on the network and calculated averages. Analyzing these figures, we see that information spreads faster using centrality measures based initial dolphins selection algorithms, and the one using estimated node betweenness achieves the best performance in these four scenarios in this network.

We can see from Figures 6.12, 6.14, and 6.15 that selecting the nodes with highest centrality values as the initial nodes to spread information has a better performance than selecting initial nodes randomly. Among all these centrality measures, node betweenness and the estimated node betweenness have the best performance for finding influential nodes and thus have a better performance for information spreading. We also notice that the more initial nodes we selected, the faster the information will be spread, which agrees with our intuition.

6.3 Spreading Information with the Help of Community Detection

With the knowledge introduced in previous two sections, let us now consider a 100-node graph, as shown in Figure 6.16, which consists of two star subgraphs connected by a bridge. Figure 6.16 displays the node indices and the corresponding node betweenness B_i's represented by the color of the nodes, where red represents high node betweenness value, while blue represents low value. Note that in this network, node 2 and node 1 have the top two node betweenness values. Suppose we want to assign a pair of nodes initially to spread information, node 1 and node 2 seems to be a very good pair. Because as was mentioned in Section 6.2, node betweenness is a very good centrality measure for identifying influential nodes who are able to efficiently spread information in the network. With these two nodes chosen as the initial step, it would

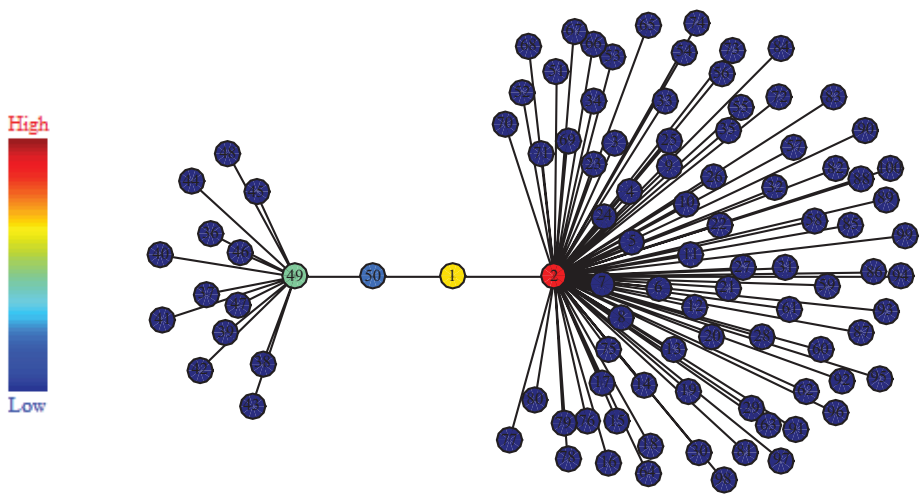


Figure 6.16: A 100-node graph with its node indices and node betweenness

take only three steps to reach everyone in this network. However, it turns out not to be the best way for this graph as we could select node 2 and node 49 instead to start the spreading, and the whole network would be reached in just one step.

Now consider the 37-node graph, as shown in Figure 6.11(a). Suppose we now select three nodes with highest node betweenness to start with, i.e., node 2, node 1 and node 10, the number of nodes could be reached within certain steps might be affected, due to the fact that the three nodes are quite close to each other and have many shared connections. Notice that there are three communities in this network, as studied in Section 6.1. If we select the nodes with the highest betweenness value from each community, would it be able to reach more nodes within certain steps or reach every node within fewer number of steps? In other words, would the knowledge of community structure in a network be helpful for finding influential people more accurately?

To test this idea, we combine the algorithms for detecting community structures and information spreading in Section 6.1 and 6.2, and present a number of tests on computer generated and real world networks. We evaluate and compare the number of nodes reached within certain steps based on the following three methods to select n initial nodes for information spreading:

Method 1 Select n nodes with the highest node betweenness in the network.

Method 2 Cluster nodes in the network into n communities using the edge betweenness based Girvan-Newman algorithm. In each community, select the node with the highest node betweenness.

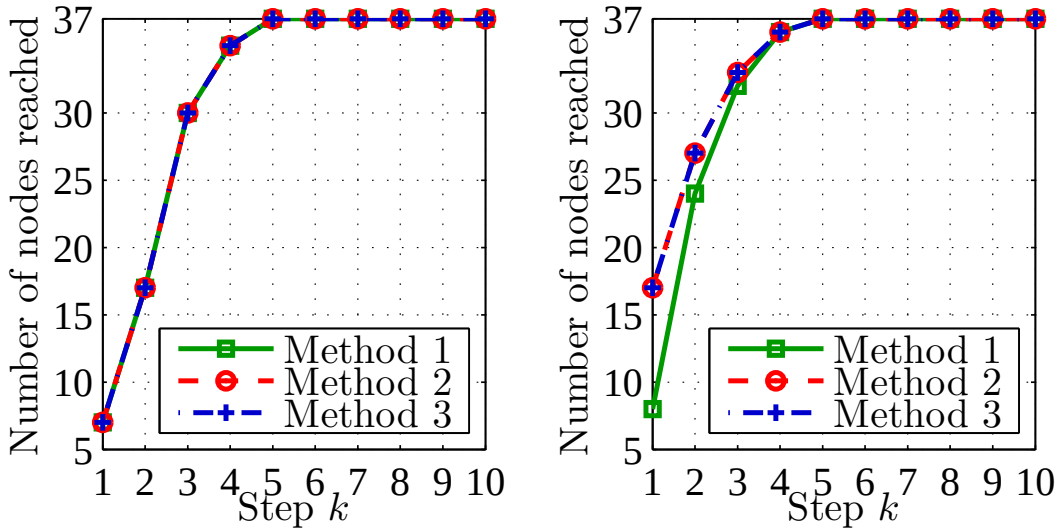
Method 3 Cluster nodes in the network into n communities using the estimated edge betweenness based Girvan-Newman algorithm. In each community, select the node with the highest estimated node betweenness.

First, we consider computer generated graphs. To test and compare the performance of three methods above, we apply them to a large set of artificial, computer generated graphs which can be clustered. Take the network in Figure 6.2 as an example, which is constructed with 37 nodes. Figure 6.17 displays the comparison and evaluation results for above three methods. The following is a quick summary:

- Figure 6.17(a) presents the number of nodes reached as a function of step k when only one node is selected initially to spread information. In this scenario, all three methods achieve the same performance, since they all selected node 2 to start the spreading.

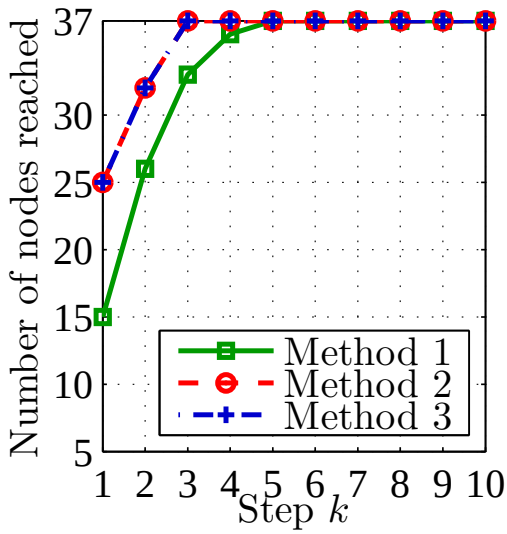
- Figure 6.17(b) plots the number of nodes reached as a function of step k when two nodes are selected initially. In this scenario, Method 1 selected node 2 and node 1 to start the spreading, while Method 2 and Method 3 selected node 2 and node 9 instead and achieve a better performance than Method 1.

- Figure 6.17(c) displays the number of nodes reached as a function of step k when three nodes are selected initially. In this scenario, Method 1 selected node 2, node 1, and node 10 to start the spreading, while Method 2 and Method 3 selected node 9, node 3, and node 6 instead and achieve a better performance than Method 1. Note that node 2 wasn't selected as previous scenarios did,

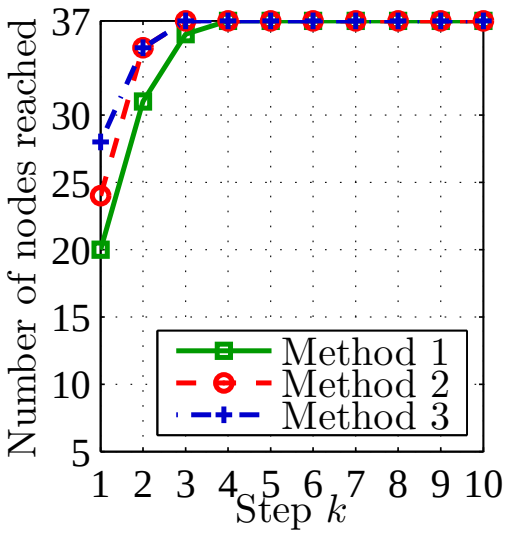


(a) $n = 1$.

(b) $n = 2$.

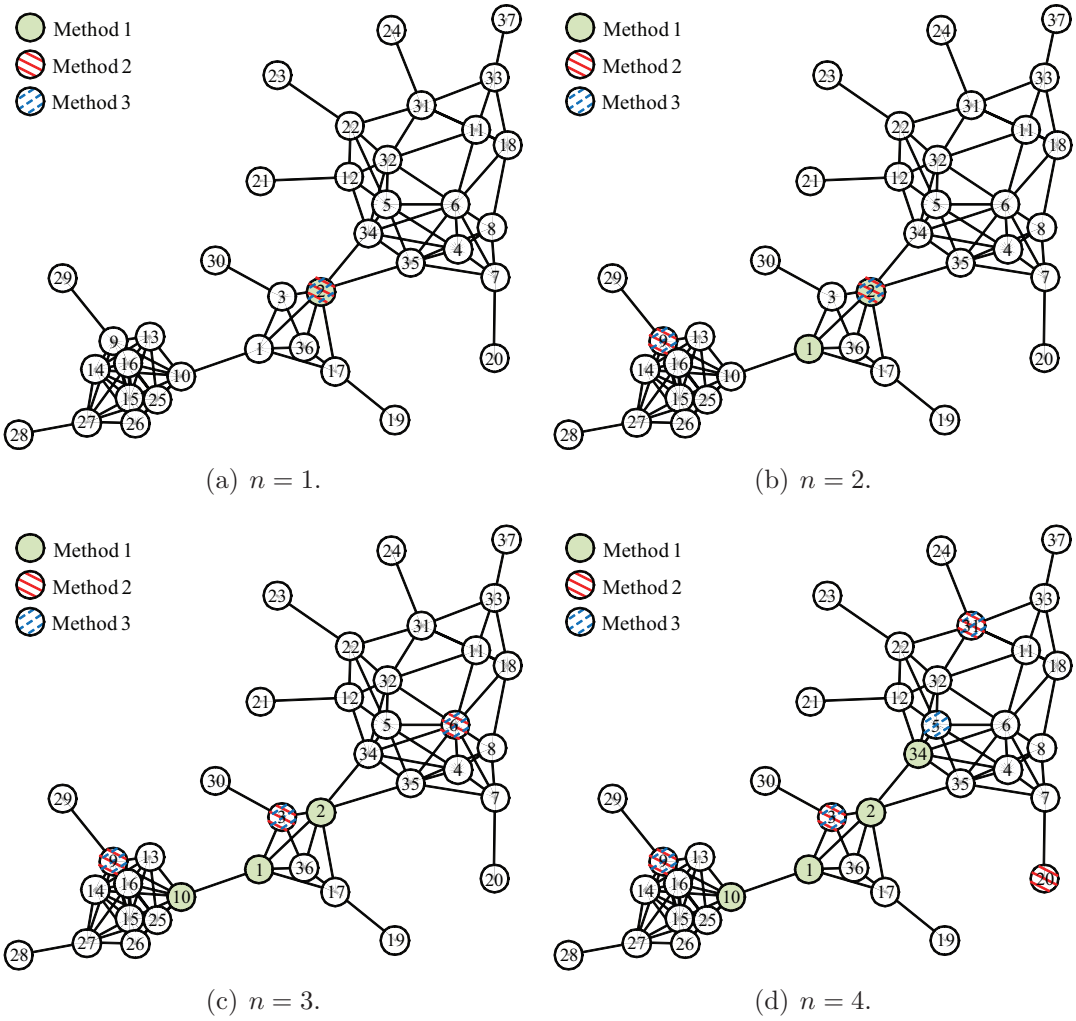


(c) $n = 3$.



(d) $n = 4$.

Figure 6.17: An comparison of the performance of Method 1–3 on a 37-node graph.



(a) $n = 1$. (b) $n = 2$.

(c) $n = 3$. (d) $n = 4$.

Figure 6.18: Initial nodes selected by Method 1–3 on a 37-node graph.

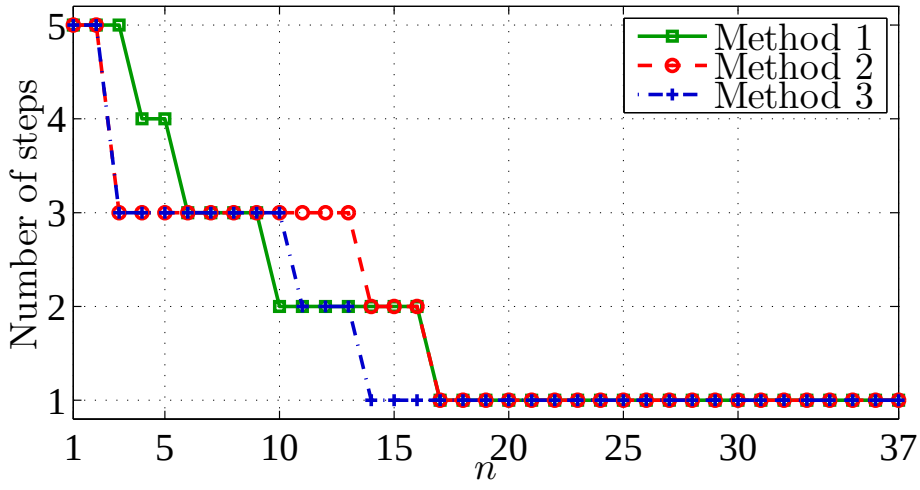


Figure 6.19: Number of steps needed to reach everyone on a 37-node graph.

since we recalculated the centrality value for all nodes when network community structure was updated.

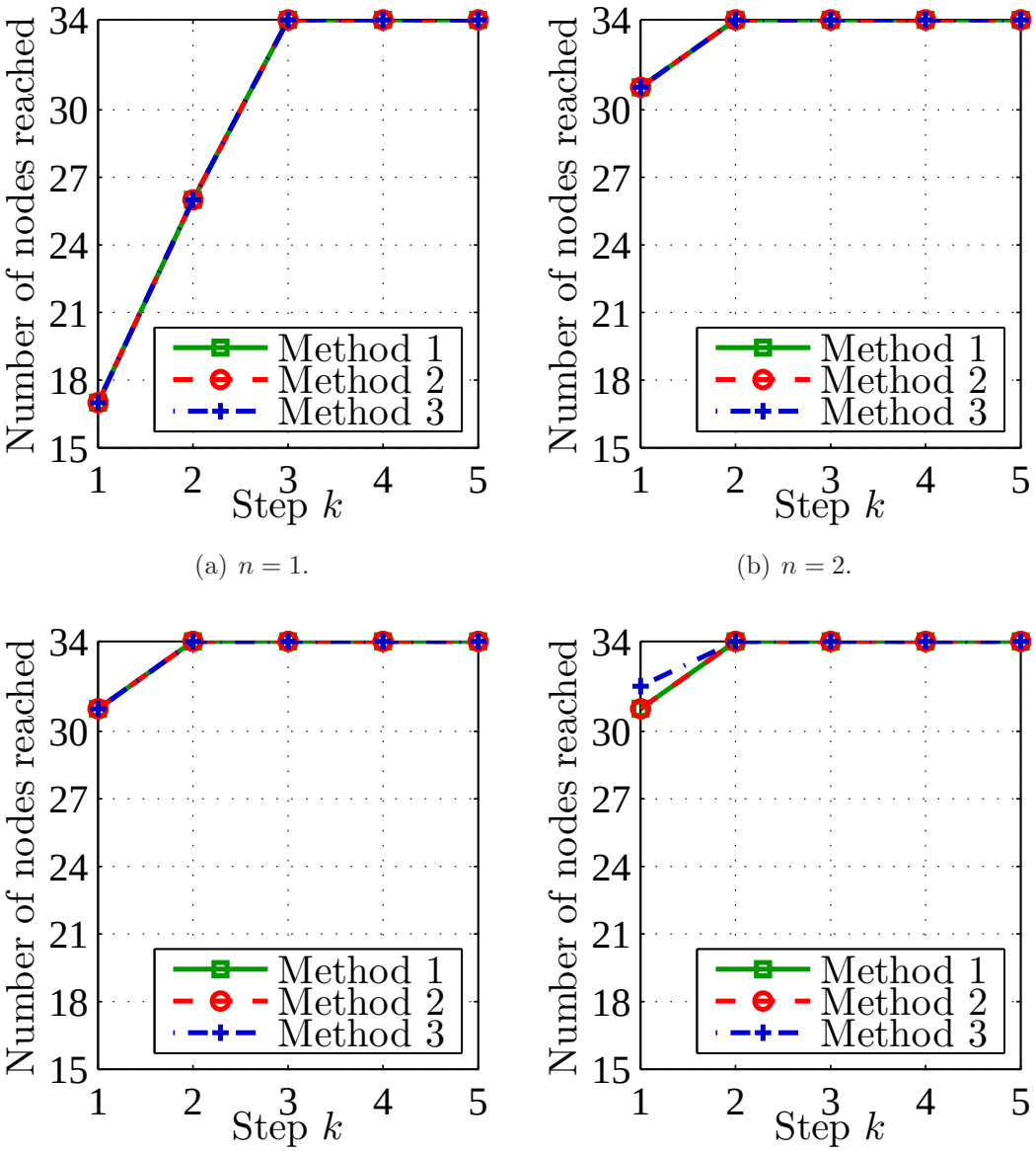
- Figure 6.17(d) shows the number of nodes reached as a function of step k when four nodes are selected initially. In this scenario, Method 1 selected node 2, node 1, node 10, and node 34 to start the spreading, Method 2 selected node 9, node 3, node 31 and node 20, while Method 3 selected node 9, node 3, node 31 and node 5 and achieves a better performance than Method 1 and Method 2.

Figure 6.18 illustrates the initial nodes selected by Method 1–3. Figure 6.19 shows the number of steps needed to reach everyone in this 37-node graph, as a function of the number of initially selected nodes n. Analyzing these figures, we see that information spreads faster using community detection based algorithms (i.e., Method 2 and Method 3) when n is very small (i.e., $n < 6$). We also notice that when n is very large (i.e.,$n \geq 17$), all the nodes can be reached within one step, all three methods achieving the same performance, which agrees with the intuition.

We next test Method 1–3 on Zachary's karate club network in Figure 6.5. Note that we also implemented the community detection and information spreading algorithms on this network in Section 6.1 and 6.2 respectively. We apply the combined algorithm based on the results obtained in these two sections. Figure 6.20 displays the comparison and evaluation results for above three methods. The following is a quick summary:

- Figure 6.20(a) presents the number of people reached as a function of step k when only one person is selected initially to spread information. In this scenario, all three methods achieve the same performance, since they all selected node 1 to start the spreading.

- Figure 6.20(b) plots the number of people reached as a function of step k when two people are selected initially. In this scenario, all three methods selected



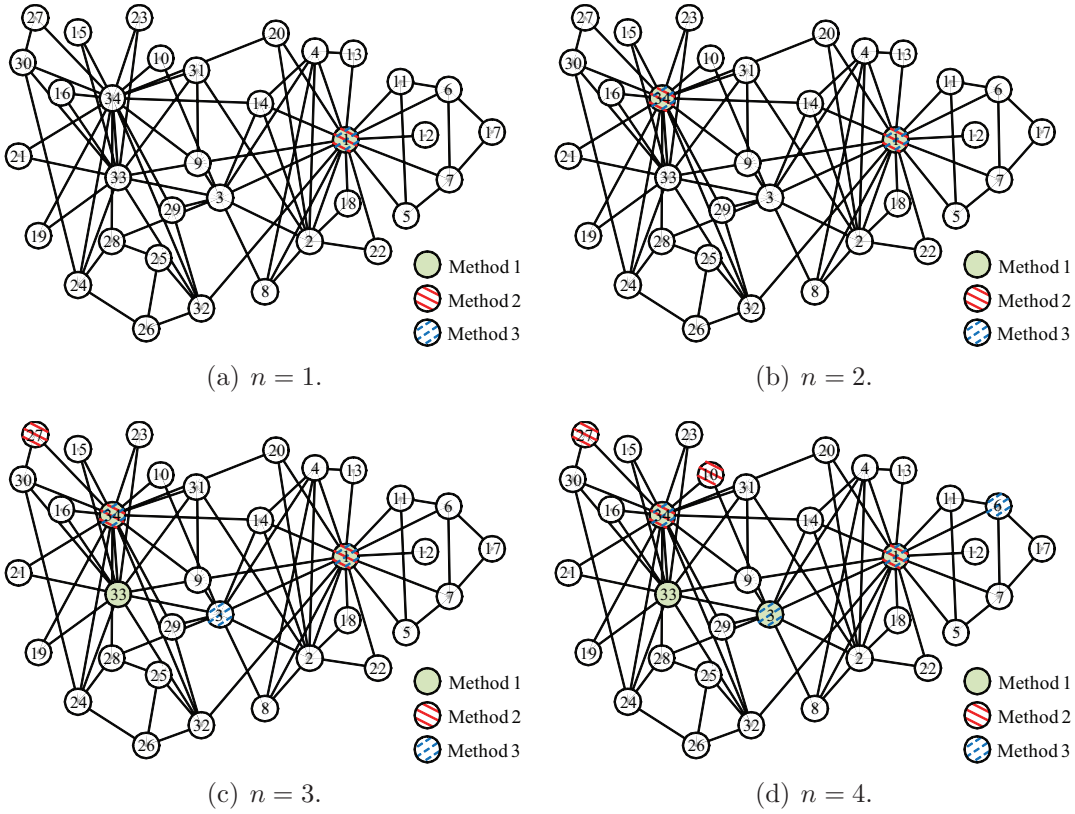
(a) $n = 1$.

(b) $n = 2$.

(c) $n = 3$.

(d) $n = 4$.

Figure 6.20: An comparison of the performance of Method 1–3 on Zachary's karate club network.



(a) $n = 1$. (b) $n = 2$.

(c) $n = 3$. (d) $n = 4$.

Figure 6.21: Initial nodes selected by Method 1–3 on Zachary's karate club network.

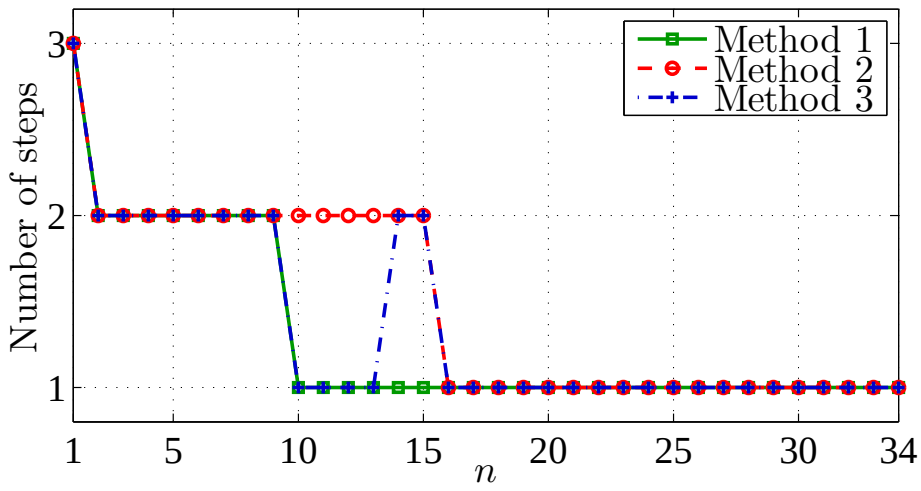


Figure 6.22: Number of steps needed to reach everyone on Zachary's karate club network.

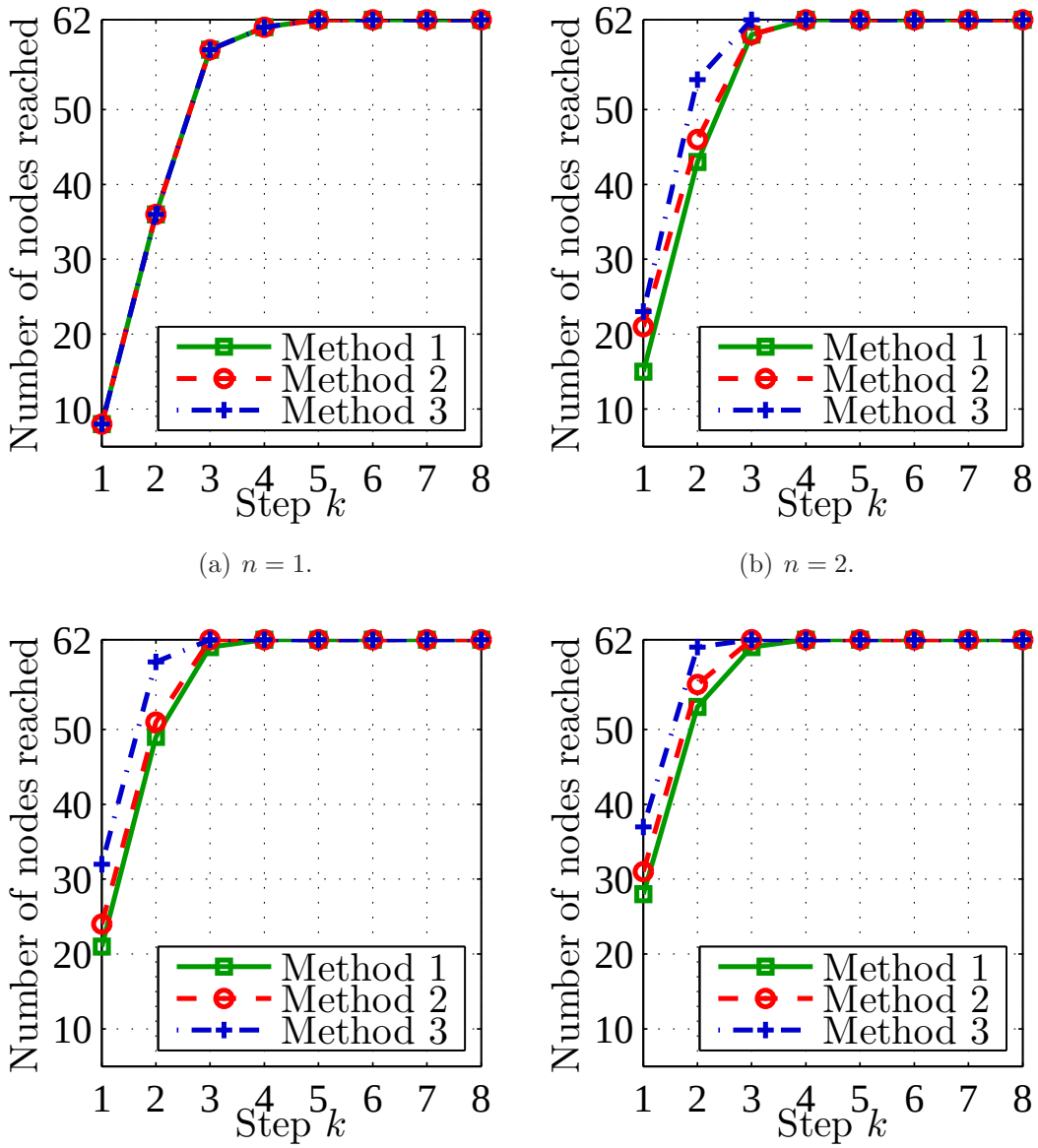
node 1 and node 34 to start the spreading, achieving the same performance.

- Figure 6.20(c) displays the number of people reached as a function of step k when three people are selected initially. In this scenario, Method 1 selected node 1, node 34 and node 33 to start the spreading, Method 2 selected node 1, node 34 and node 27, and Method 3 selected node 1, node 34 and node 3. All three methods achieve the same performance.

- Figure 6.20(d) shows the number of people reached as a function of step k when four people are selected initially. In this scenario, Method 1 selected node 1, node 34, node 33, and node 3 to start the spreading, Method 2 selected node 1, node 34, node 27 and node 10, and Method 3 selected node 1, node 34, node 3, and node 6, achieving slightly better performance than other methods.

Figure 6.21 illustrates the initial nodes selected by Method 1–3. Figure 6.22 shows the number of steps needed to reach everyone in Zachary's karate club network, as a function of the number of initially selected nodes n. Analyzing these figures, we see that information spreads at around the same speed based on three methods when n is very small (i.e., $n \leq 10$) and very large (i.e., $n > 15$).

We finally test three methods on the dolphin social network. Again, we implemented the community detection and information spreading algorithms on this network in Section 6.1 and 6.2 respectively. We apply the combined algorithms based on the results obtained in these two sections. Figure 6.23 displays the comparison and evaluation results for above three methods. Specifically, Figure 6.23(a) presents the number of dolphins reached as a function of step k when only one dolphin is selected initially to spread information. In this scenario, all three methods achieve the same performance, since they all selected the same node to start the spreading. Likewise, Figure 6.23(b) plots the number of dolphins reached as a function of step k when two dolphins are selected initially. Figure 6.23(c) displays the number of dolphins reached



(a) $n = 1$. (b) $n = 2$.

(c) $n = 3$. (d) $n = 4$.

Figure 6.23: An comparison of the performance of Method 1–3 on the dolphin social network.

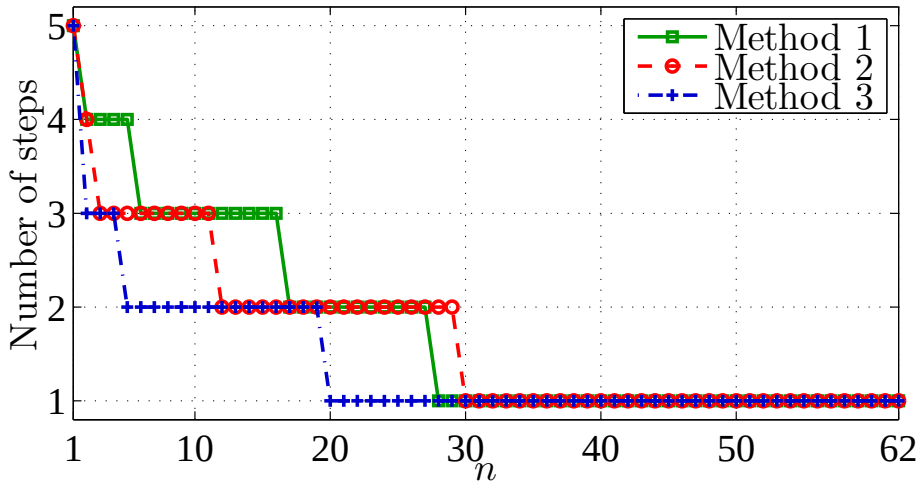


Figure 6.24: Number of steps needed to reach everyone on the dolphin social network.

as a function of step k when three dolphins are selected initially. Figure 6.23(d) shows the number of dolphins reached as a function of step k when four dolphins are selected initially. In the last three scenarios shown in (b)–(d), Method 3 achieves the best performance and Method 1 results in the worst performance. Figure 6.24 shows the number of steps needed to reach everyone in the dolphin social network as a function of the number of initially selected nodes n. Analyzing these figures, we see that information spreads faster using community detection based algorithms (i.e., Method 2 and Method 3) when n is small (i.e., $n < 18$), and Method 3 performs no worse than the other two methods all the time, and perform the best when n is small (i.e., $n < 12$) and when n is approaching to the half size of the network (i.e., $20 \leq n < 30$). We also notice that when n is very large (i.e., $n \geq 30$), all the nodes can be reached within one step, all three methods achieving the same performance, which agrees with the intuition.

Finally, note from Figures 6.17–6.24 that with the knowledge of community structures, information could spread faster in the network by correctly detecting influential people, and the estimated betweenness based method (i.e., Method 3) performs very well in the almost all scenarios, especially when the number of initially selected nodes

n is small.

6.4 Conclusion

In this chapter, we have experimented our algorithms proposed in Chapter 4 for estimating node and edge betweenness centralities on both computer generated graphs and real networks for community detection—based on Girvan-Newman algorithm [122]—and information spreading. We have also proposed a method for spreading information with the knowledge of community structures. The evaluation results indicate that the estimated edge betweenness performs very well in correctly detecting community structures especially when the number of communities is very small compared with the size of the network. Also, the estimated node betweenness has a pretty good performance for identifying influential nodes. Moreover, the evaluation results have also shown that with the knowledge of community structures, information could spread faster in the network by correctly detecting influential people, and the method using estimated betweenness performs very well in almost all scenarios. This is especially true when we have a limited number of initial people to select.

Chapter 7

Conclusions

7.1 Overall Summary

In this dissertation, we have developed a novel collection of scalable distributed algorithms, which enable nodes in a large-scale network to cooperatively learn how important they individually are, with only local interaction and with neither global coordination nor knowledge of the network topology. These algorithms use the most fundamental centrality measures from the area of complex networks, namely, the betweenness centrality, closeness centrality, as well as a subset of their variations.

First, we have introduced a set of continuous- and discrete-time distributed algorithms, which enable nodes in a tree graph to cooperatively compute their individual node betweenness and incident edge betweenness. Constructed using a dynamical systems approach, we have shown that the algorithms possess several positive attributes, such as being simple, scalable, and exponentially or finite-time stable with strong convergence characteristics, and being applicable to time-varying tree graphs.

Then, we have designed and analyzed continuous- and discrete-time distributed algorithms that allow nodes in a tree graph to cooperatively determine their classic and exponential closeness using an algebraic-relationships-turned-dynamical-systems

approach.

Next, we have developed a scalable distributed algorithm that enables nodes in a general graph to cooperatively estimate their individual betweenness and the betweenness of edges incident on them with only local interaction and without any centralized coordination, nor high memory usages. To arrive at this algorithm, we have exploited various local properties of shortest paths and considered an unconstrained distributed optimization problem. We have also shown via simulation on a number of random geometric graphs that the algorithm is fairly accurate in terms of ordering, but has room for improvement.

Moreover, we have developed a scalable distributed algorithm that enables nodes in a general graph to cooperatively estimate their individual closeness with only local interaction and without any centralized coordination, nor high memory usages. We have also shown via extensive simulation on three common random graphs that this algorithm is fairly accurate, but has room for improvement.

Finally, we have experimented our algorithms for estimating node and edge betweenness centralities on both computer generated graphs and real networks for community detection—based on Girvan-Newman algorithm—and information spreading. We have also proposed a method for better spreading information with the knowledge of community structures. The method using estimated betweenness performs very well in almost all scenarios, especially when we have a limited number of initial people to select.

7.2 Future Work

Although this dissertation has developed a novel collection of simple and scalable distributed algorithms for computing and estimating betweenness and closeness centrality measures, there are several possible future research directions, which may be

of interest:

- extending the resulting algorithms from Chapters 2–5, so that they are applicable to directed and weighted networks with time-varying topologies (to account for directed and weighted edges, some of the measures will need to be appropriately redefined);

- deriving additional constraints using variables $x_{(i,j)}$'s and F_i's introduced in Chapters 4 and 5 that decrease the size of the feasible set, and incorporating them into the algorithms from Chapters 4 and 5, to increase the accuracy of the estimation;

- deriving additional constraints using other candidate variables whose values depend on the shortest paths;

  More specifically, for example, we could define the following shortest-path-dependent variables: for each $i \in \mathcal{V}$ and $j \in \mathcal{N}_i$, let

$$\langle i,j\rangle] \triangleq \sum_{\substack{k\in\mathcal{V}\\k\neq i}} \sum_{\substack{\ell\in\mathcal{V}\\\ell\neq i,k}} \frac{\sigma(k,\ell,(i,j))}{\sigma(k,\ell)}, \tag{7.1}$$

$$[i,j\rangle] \triangleq \sum_{\substack{\ell\in\mathcal{V}\\\ell\neq i}} \frac{\sigma(i,\ell,(i,j))}{\sigma(i,\ell)}, \tag{7.2}$$

$$[\langle i,j\rangle] \triangleq \sum_{k\in\mathcal{V}} \sum_{\substack{\ell\in\mathcal{V}\\\ell\neq i,k}} \frac{\sigma(k,\ell,(i,j))}{\sigma(k,\ell)}, \tag{7.3}$$

  so that, roughly speaking, $\langle i,j\rangle]$ represents the fraction of shortest paths that go through edge (i,j) but do not begin at node i; $[i,j\rangle]$ represents the fraction of shortest paths that go through edge (i,j) and begin at node i; and $[\langle i,j\rangle]$ represents the fraction of shortest paths that go through edge (i,j).

  Given the above variables, a possible next step is to try to derive enough number of equations and inequalities relating them and connecting them to node

betweenness B_i and edge betweenness $B_{\{i,j\}}$. To date, we have obtained the following results, which are summarized in the lemma below:

Lemma 15. *The node betweenness B_i and edge betweenness $B_{\{i,j\}}$ satisfy*

$$\frac{1}{2}\sum_{j\in\mathcal{N}_i} B_{\{i,j\}} = B_i + N - 1, \quad \forall i \in \mathcal{V}\ \forall j \in \mathcal{N}_i.$$

In addition, they satisfy

$$2(B_i - B_j) = \sum_{h\in\mathcal{N}_i} B_{\{i,h\}} - \sum_{k\in\mathcal{N}_j} B_{\{j,k\}}, \quad \forall i \in \mathcal{V}\ \forall j \in \mathcal{V}.$$

Proof. Based on (1.1), (2.1), and (7.1)–(7.3), $\forall i \in \mathcal{V}$ we have

$$\begin{aligned}\frac{1}{2}\sum_{j\in\mathcal{N}_i} B_{\{i,j\}} &= \sum_{j\in\mathcal{N}_i}[\langle i,j\rangle] = \sum_{j\in\mathcal{N}_i}\langle i,j\rangle] + \sum_{j\in\mathcal{N}_i}[i,j\rangle] \\ &= B_i + N - 1.\end{aligned}$$

The second equation is a consequence of the first. □

Lemma 15 suggests that, if a distributed algorithm for computing edge betweenness $B_{\{i,j\}}$ on general graphs can be developed, the algorithm could readily be used to compute node betweenness B_i, thereby solving two problems at once. Thus, possible future research could continue along this line of effort to relate $\langle i,j\rangle]$, $[i,j\rangle]$, and $[\langle i,j\rangle]$, leading ultimately to a distributed algorithms for solving the problem.

- developing distributed algorithms for computing other centrality measures on general graphs, building perhaps on the ideas of this dissertation;
- finding out more applications besides the ones mentioned in Chapter 6, and testing our algorithms on more real networks.

Bibliography

[1] "The Opte Project," http://www.opte.org.

[2] M. Franceschet, "Bottlenose dolphins," 2016, https://users.dimi.uniud.it/ massimo.franceschet/bottlenose/bottlenose.html.

[3] M. E. J. Newman, *Networks: An Introduction*. New York, NY: Oxford University Press, 2010.

[4] R. Cohen and S. Havlin, *Complex Networks: Structure, Robustness and Function*. New York, NY: Cambridge University Press, 2010.

[5] S. P. Borgatti, "Centrality and network flow," *Social Networks*, vol. 27, no. 1, pp. 55–71, 2005.

[6] J. M. Anthonisse, "The rush in a directed graph," Stichting Mathematisch Centrum, Amsterdam, Netherlands, Technical Report BN 9/71, 1971.

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

[8] S. Jin, Z. Huang, Y. Chen, D. Chavarría-Miranda, J. Feo, and P. C. Wong, "A novel application of parallel betweenness centrality to power grid contingency analysis," in *Proc. IEEE International Symposium on Parallel and Distributed Processing*, Atlanta, GA, 2010, pp. 1–7.

[9] S. Lämmer, B. Gehlsen, and D. Helbing, "Scaling laws in the spatial structure of urban road networks," *Physica A: Statistical Mechanics and Its Applications*, vol. 363, no. 1, pp. 89–95, 2006.

[10] A. del Sol, H. Fujihashi, and P. O'Meara, "Topology of small-world networks of protein-protein complex structures," *Bioinformatics*, vol. 21, no. 8, pp. 1311–1315, 2005.

[11] L. Leydesdorff, "Betweenness centrality as an indicator of the interdisciplinarity of scientific journals," *Journal of the American Society for Information Science and Technology*, vol. 58, no. 9, pp. 1303–1319, 2007.

[12] A. Bavelas, "Communication patterns in task-oriented groups," *Journal of the Acoustical Society of America*, vol. 22, no. 6, pp. 725–730, 1950.

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

[14] S. P. Borgatti, "Centrality and AIDS," *Connections*, vol. 18, no. 1, pp. 111–113, 1995.

[15] K. Okamoto, W. Chen, and X.-Y. Li, "Ranking of closeness centrality for large-scale social networks," in *Frontiers in Algorithmics, Lecture Notes in Computer Science*, F. P. Preparata, X. Wu, and J. Yin, Eds. Berlin, Germany: Springer, 2008, pp. 186–195.

[16] Z. Wang, A. Scaglione, and R. J. Thomas, "Electrical centrality measures for electric power grid vulnerability analysis," in *Proc. IEEE Conference on Decision and Control*, Atlanta, GA, 2010, pp. 5792–5797.

[17] A. B. M. Nasiruzzaman, H. R. Pota, and M. A. Mahmud, "Application of centrality measures of complex network framework in power grid," in *Proc. Conference of the IEEE Industrial Electronics Society*, Melbourne, Australia, 2011, pp. 4660–4665.

[18] R. W. Floyd, "Algorithm 97: Shortest path," *Communications of the ACM*, vol. 5, no. 6, p. 345, 1962.

[19] D. B. Johnson, "A note on Dijkstra's shortest path algorithm," *Journal of the ACM*, vol. 20, no. 3, pp. 385–388, 1973.

[20] L. C. Freeman, S. P. Borgatti, and D. R. White, "Centrality in valued graphs: A measure of betweenness based on network flow," *Social Networks*, vol. 13, no. 2, pp. 141–154, 1991.

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

[22] 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.

[23] K. A. Lehmann and M. Kaufmann, "Decentralized algorithms for evaluating centrality in complex networks," University of Tübingen, Tübingen, Germany, Technical Report, 2003.

[24] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, 2004.

[25] J. Cortés, "Finite-time convergent gradient flows with applications to network consensus," *Automatica*, vol. 42, no. 11, pp. 1993–2000, 2006.

[26] A. Tahbaz-Salehi and A. Jadbabaie, "Small world phenomenon, rapidly mixing Markov chains, and average consensus algorithms," in *Proc. IEEE Conference on Decision and Control*, New Orleans, LA, 2007, pp. 276–281.

[27] R. Olfati-Saber, "Ultrafast consensus in small-world networks," in *Proc. American Control Conference*, Portland, OR, 2005, pp. 2371–2378.

[28] J. Wang and N. Elia, "Consensus over networks with dynamic channels," in *Proc. American Control Conference*, Seattle, WA, 2008, pp. 2637–2642.

[29] Y. Hatano and M. Mesbahi, "Agreement over random networks," *IEEE Transactions on Automatic Control*, vol. 50, no. 11, pp. 1867–1872, 2005.

[30] W. Ren and R. W. Beard, "Consensus seeking in multiagent systems under dynamically changing interaction topologies," *IEEE Transactions on Automatic Control*, vol. 50, no. 5, pp. 655–661, 2005.

[31] A. Tahbaz-Salehi and A. Jadbabaie, "Necessary and sufficient conditions for consensus over random independent and identically distributed switching graphs," in *Proc. IEEE Conference on Decision and Control*, New Orleans, LA, 2007, pp. 4209–4214.

[32] S. Di Cairano, A. Pasini, A. Bemporad, and R. M. Murray, "Convergence properties of dynamic agents consensus networks with broken links," in *Proc. American Control Conference*, Seattle, WA, 2008, pp. 1362–1367.

[33] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[34] R. Olfati-Saber and J. S. Shamma, "Consensus filters for sensor networks and distributed sensor fusion," in *Proc. IEEE Conference on Decision and Control*, Seville, Spain, 2005, pp. 6698–6703.

[35] D. P. Spanos, R. Olfati-Saber, and R. M. Murray, "Dynamic consensus for mobile networks," in *Proc. IFAC World Congress*, Prague, Czech Republic, 2005.

[36] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *Proc. IEEE Symposium on Foundations of Computer Science*, Cambridge, MA, 2003, pp. 482–491.

[37] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems & Control Letters*, vol. 53, no. 1, pp. 65–78, 2004.

[38] D. S. Scherber and H. C. Papadopoulos, "Distributed computation of averages over ad hoc networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 4, pp. 776–787, 2005.

[39] D. B. Kingston and R. W. Beard, "Discrete-time average-consensus under switching network topologies," in *Proc. American Control Conference*, Minneapolis, MN, 2006, pp. 3551–3556.

[40] A. Olshevsky and J. N. Tsitsiklis, "Convergence rates in distributed consensus and averaging," in *Proc. IEEE Conference on Decision and Control*, San Diego, CA, 2006, pp. 3387–3392.

[41] L. Xiao, S. Boyd, and S.-J. Kim, "Distributed average consensus with least-mean-square deviation," *Journal of Parallel and Distributed Computing*, vol. 67, no. 1, pp. 33–46, 2007.

[42] F. Fagnani and S. Zampieri, "Randomized consensus algorithms over large scale networks," *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 4, pp. 634–649, 2008.

[43] M. Zhu and S. Martínez, "Dynamic average consensus on synchronous communication networks," in *Proc. American Control Conference*, Seattle, WA, 2008, pp. 4382–4387.

[44] A. Olshevsky and J. N. Tsitsiklis, "Convergence speed in distributed consensus and averaging," *SIAM Journal on Control and Optimization*, vol. 48, no. 1, pp. 33–55, 2009.

[45] B. N. Oreshkin, M. J. Coates, and M. G. Rabbat, "Optimization and analysis of distributed averaging with short node memory," *IEEE Transactions on Signal Processing*, vol. 58, no. 5, pp. 2850–2865, 2010.

[46] M. H. DeGroot, "Reaching a consensus," *Journal of the American Statistical Association*, vol. 69, no. 345, pp. 118–121, 1974.

[47] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, 2003.

[48] M. Cao, D. A. Spielman, and A. S. Morse, "A lower bound on convergence of a distributed network consensus algorithm," in *Proc. IEEE Conference on Decision and Control*, Seville, Spain, 2005, pp. 2356–2361.

[49] L. Moreau, "Stability of multiagent systems with time-dependent communication links," *IEEE Transactions on Automatic Control*, vol. 50, no. 2, pp. 169–182, 2005.

[50] S. Sundaram and C. N. Hadjicostis, "Finite-time distributed consensus in graphs with time-invariant topologies," in *Proc. American Control Conference*, New York, NY, 2007, pp. 711–716.

[51] A. Olshevsky and J. N. Tsitsiklis, "On the nonexistence of quadratic Lyapunov functions for consensus algorithms," *IEEE Transactions on Automatic Control*, vol. 53, no. 11, pp. 2642–2645, 2008.

[52] A. Tahbaz-Salehi and A. Jadbabaie, "Consensus over ergodic stationary graph processes," *IEEE Transactions on Automatic Control*, vol. 55, no. 1, pp. 225–230, 2010.

[53] A. Nedić, A. Ozdaglar, and P. A. Parrilo, "Constrained consensus and optimization in multi-agent networks," *IEEE Transactions on Automatic Control*, vol. 55, no. 4, pp. 922–938, 2010.

[54] J. N. Tsitsiklis, "Problems in decentralized decision making and computation," Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, 1984.

[55] M. Jelasity and A. Montresor, "Epidemic-style proactive aggregation in large overlay networks," in *Proc. IEEE International Conference on Distributed Computing Systems*, Tokyo, Japan, 2004, pp. 102–109.

[56] A. Montresor, M. Jelasity, and O. Babaoglu, "Robust aggregation protocols for large-scale overlay networks," in *Proc. IEEE/IFIP International Conference on Dependable Systems and Networks*, Florence, Italy, 2004, pp. 19–28.

[57] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2508–2530, 2006.

[58] M. Cao, D. A. Spielman, and E. M. Yeh, "Accelerated gossip algorithms for distributed computation," in *Proc. Allerton Conference on Communication, Control, and Computing*, Monticello, IL, 2006, pp. 952–959.

[59] J.-Y. Chen, G. Pandurangan, and D. Xu, "Robust computation of aggregates in wireless sensor networks: Distributed randomized algorithms and analysis," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 9, pp. 987–1000, 2006.

[60] C. C. Moallemi and B. Van Roy, "Consensus propagation," *IEEE Transactions on Information Theory*, vol. 52, no. 11, pp. 4753–4766, 2006.

[61] M. Mehyar, D. Spanos, J. Pongsajapan, S. H. Low, and R. M. Murray, "Asynchronous distributed averaging on communication networks," *IEEE/ACM Transactions on Networking*, vol. 15, no. 3, pp. 512–520, 2007.

[62] V. Borkar and P. P. Varaiya, "Asymptotic agreement in distributed estimation," *IEEE Transactions on Automatic Control*, vol. 27, no. 3, pp. 650–655, 1982.

[63] J. N. Tsitsiklis and M. Athans, "Convergence and asymptotic agreement in distributed decision problems," *IEEE Transactions on Automatic Control*, vol. 29, no. 1, pp. 42–50, 1984.

[64] J. N. Tsitsiklis, D. P. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Transactions on Automatic Control*, vol. 31, no. 9, pp. 803–812, 1986.

[65] V. D. Blondel, J. M. Hendrickx, A. Olshevsky, and J. N. Tsitsiklis, "Convergence in multiagent coordination, consensus, and flocking," in *Proc. IEEE Conference on Decision and Control*, Seville, Spain, 2005, pp. 2996–3000.

[66] M. Cao, A. S. Morse, and B. D. O. Anderson, "Coordination of an asynchronous multi-agent system via averaging," in *Proc. IFAC World Congress*, Prague, Czech Republic, 2005.

[67] L. Fang and P. J. Antsaklis, "Information consensus of asynchronous discrete-time multi-agent systems," in *Proc. American Control Conference*, Portland, OR, 2005, pp. 1883–1888.

[68] L. Fang, P. J. Antsaklis, and A. Tzimas, "Asynchronous consensus protocols: Preliminary results, simulations and open questions," in *Proc. IEEE Conference on Decision and Control*, Seville, Spain, 2005, pp. 2194–2199.

[69] L. Fang and P. J. Antsaklis, "On communication requirements for multi-agent consensus seeking," in *Networked Embedded Sensing and Control*, ser. Lecture Notes in Control and Information Sciences, P. J. Antsaklis and P. Tabuada, Eds. Berlin, Germany: Springer-Verlag, 2006, vol. 331, pp. 53–67.

[70] C. Y. Tang and J. Lu, "Controlled hopwise averaging: Bandwidth/energy-efficient asynchronous distributed averaging for wireless networks," in *Proc. American Control Conference*, St. Louis, MO, 2009, pp. 1561–1568.

[71] J. Lu and C. Y. Tang, "Convergence rate of controlled hopwise averaging on various graphs," in *Proc. IEEE Conference on Decision and Control*, Orlando, FL, 2011, pp. 4290–4295.

[72] ——, "Controlled hopwise averaging and its convergence rate," *IEEE Transactions on Automatic Control*, vol. 57, no. 4, pp. 1005–1012, 2012.

[73] S. Martínez, J. Cortés, and F. Bullo, "Motion coordination with distributed information," *IEEE Control Systems Magazine*, vol. 27, no. 4, pp. 75–88, 2007.

[74] J. Fax and R. Murray, "Information flow and cooperative control of vehicle formations," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1465–1476, 2004.

[75] W. Ren, R. W. Beard, and E. M. Atkins, "Information consensus in multivehicle cooperative control," *IEEE Control Systems Magazine*, vol. 27, no. 2, pp. 71–82, 2007.

[76] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Transactions on Automatic Control*, vol. 51, no. 3, pp. 401–420, 2006.

[77] H. G. Tanner, A. Jadbabaie, and G. J. Pappas, "Flocking in fixed and switching networks," *IEEE Transactions on Automatic Control*, vol. 52, no. 5, pp. 863–868, 2007.

[78] D. P. Spanos, R. Olfati-Saber, and R. M. Murray, "Distributed sensor fusion using dynamic consensus," in *Proc. IFAC World Congress*, Prague, Czech Republic, 2005.

[79] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *Proc. International Symposium on Information Processing in Sensor Networks*, Los Angeles, CA, 2005, pp. 63–70.

[80] ——, "A space-time diffusion scheme for peer-to-peer least-squares estimation," in *Proc. International Conference on Information Processing in Sensor Networks*, Nashville, TN, 2006, pp. 168–176.

[81] J. Lu and C. Y. Tang, "Distributed asynchronous algorithms for solving positive definite linear equations over networks—Part I: Agent networks," in *Proc. IFAC Workshop on Estimation and Control of Networked Systems*, Venice, Italy, 2009, pp. 252–257.

[82] ——, "Distributed asynchronous algorithms for solving positive definite linear equations over networks—Part II: Wireless networks," in *Proc. IFAC Workshop on Estimation and Control of Networked Systems*, Venice, Italy, 2009, pp. 258–263.

[83] J. Cortés, "Distributed algorithms for reaching consensus on general functions," *Automatica*, vol. 44, no. 3, pp. 726–737, 2008.

[84] D. Mosk-Aoyama and D. Shah, "Computing separable functions via gossip," in *Proc. ACM Symposium on Principles of Distributed Computing*, Denver, CO, 2006, pp. 113–122.

[85] A. Tahbaz-Salehi and A. Jadbabaie, "A one-parameter family of distributed consensus algorithms with boundary: From shortest paths to mean hitting times," in *Proc. IEEE Conference on Decision and Control*, San Diego, CA, 2006, pp. 4664–4669.

[86] D. Bauso, L. Giarré, and R. Pesenti, "Non-linear protocols for optimal distributed consensus in networks of dynamic agents," *Systems & Control Letters*, vol. 55, no. 11, pp. 918–928, 2006.

[87] R. Olfati-Saber, "Distributed Kalman filter with embedded consensus filters," in *Proc. IEEE Conference on Decision and Control*, Seville, Spain, 2005, pp. 8179–8184.

[88] D. P. Spanos, R. Olfati-Saber, and R. M. Murray, "Approximate distributed Kalman filtering in sensor networks with quantifiable performance," in *Proc. International Symposium on Information Processing in Sensor Networks*, Los Angeles, CA, 2005, pp. 133–139.

[89] R. Olfati-Saber, "Distributed Kalman filtering for sensor networks," in *Proc. IEEE Conference on Decision and Control*, New Orleans, LA, 2007, pp. 5492–5498.

[90] S. Roy, A. Saberi, and K. Herlugson, "A control-theoretic perspective on the design of distributed agreement protocols," in *Proc. American Control Conference*, Portland, OR, 2005, pp. 1672–1679.

[91] S. Roy, K. Herlugson, and A. Saberi, "A control-theoretic approach to distributed discrete-valued decision-making in networks of sensing agents," *IEEE Transactions on Mobile Computing*, vol. 5, no. 8, pp. 945–957, 2006.

[92] S. Sundaram and C. N. Hadjicostis, "Distributed consensus and linear functional calculation in networks: An observability perspective," in *Proc. International Conference on Information Processing in Sensor Networks*, Cambridge, MA, 2007, pp. 99–108.

[93] J. M. Hendrickx, A. Olshevsky, and J. N. Tsitsiklis, "Distributed anonymous discrete function computation," *IEEE Transactions on Automatic Control*, vol. 56, no. 10, pp. 2276–2289, 2011.

[94] A. Nedić and D. P. Bertsekas, "Incremental subgradient methods for nondifferentiable optimization," *SIAM Journal on Optimization*, vol. 12, no. 1, pp. 109–138, 2001.

[95] A. Nedić, D. P. Bertsekas, and V. S. Borkar, "Distributed asynchronous incremental subgradient methods," in *Inherently Parallel Algorithms in Feasibility and Optimization and Their Applications*, D. Butnariu, Y. Censor, and S. Reich, Eds. Amsterdam, Holland: Elsevier, 2001, pp. 381–407.

[96] A. Nedić and D. P. Bertsekas, "Convergence rate of incremental subgradient algorithms," in *Stochastic Optimization: Algorithms and Applications*, S. P. Uryasev and P. M. Pardalos, Eds. Norwell, MA: Kluwer Academic Publishers, 2001, pp. 223–264.

[97] M. G. Rabbat and R. D. Nowak, "Distributed optimization in sensor networks," in *Proc. International Symposium on Information Processing in Sensor Networks*, Berkeley, CA, 2004, pp. 20–27.

[98] ——, "Quantized incremental algorithms for distributed optimization," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 4, pp. 798–808, 2005.

[99] S.-H. Son, M. Chiang, S. R. Kulkarni, and S. C. Schwartz, "The value of clustering in distributed estimation for sensor networks," in *Proc. International Conference on Wireless Networks, Communications and Mobile Computing*, Maui, HI, 2005, pp. 969–974.

[100] B. Johansson, M. Rabi, and M. Johansson, "A simple peer-to-peer algorithm for distributed optimization in sensor networks," in *Proc. IEEE Conference on Decision and Control*, New Orleans, LA, 2007, pp. 4705–4710.

[101] S. S. Ram, A. Nedić, and V. V. Veeravalli, "Stochastic incremental gradient descent for estimation in sensor networks," in *Proc. Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, 2007, pp. 582–586.

[102] ——, "Incremental stochastic subgradient algorithms for convex optimization," *SIAM Journal on Optimization*, vol. 20, no. 2, pp. 691–717, 2009.

[103] A. Nedić and A. Ozdaglar, "On the rate of convergence of distributed subgradient methods for multi-agent optimization," in *Proc. IEEE Conference on Decision and Control*, New Orleans, LA, 2007, pp. 4711–4716.

[104] B. Johansson, T. Keviczky, M. Johansson, and K. H. Johansson, "Subgradient methods and consensus algorithms for solving convex optimization problems," in *Proc. IEEE Conference on Decision and Control*, Cancun, Mexico, 2008, pp. 4185–4190.

[105] I. Lobel and A. Ozdaglar, "Convergence analysis of distributed subgradient methods over random networks," in *Proc. Allerton Conference on Communication, Control, and Computing*, Monticello, IL, 2008, pp. 353–360.

[106] A. Nedić, A. Olshevsky, A. Ozdaglar, and J. N. Tsitsiklis, "Distributed subgradient methods and quantization effects," in *Proc. IEEE Conference on Decision and Control*, Cancun, Mexico, 2008, pp. 4177–4184.

[107] A. Nedić and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[108] S. S. Ram, A. Nedić, and V. V. Veeravalli, "Asynchronous gossip algorithms for stochastic optimization," in *Proc. IEEE Conference on Decision and Control*, Shanghai, China, 2009, pp. 3581–3586.

[109] ——, "Distributed stochastic subgradient projection algorithms for convex optimization," *Journal of Optimization Theory and Applications*, vol. 147, no. 3, pp. 516–545, 2010.

[110] M. Zhu and S. Martínez, "On distributed convex optimization under inequality and equality constraints," *IEEE Transactions on Automatic Control*, vol. 57, no. 1, pp. 151–164, 2012.

[111] T. A. Davis and Y. Hu, "The university of florida sparse matrix collection," *ACM Transactions on Mathematical Software (TOMS)*, vol. 38, no. 1, p. 1, 2011.

[112] M. E. J. Newman, "A measure of betweenness centrality based on random walks," *Social Networks*, vol. 27, no. 1, pp. 39–54, 2005.

[113] C. Dangalchev, "Residual closeness in networks," *Physica A: Statistical Mechanics and Its Applications*, vol. 365, no. 2, pp. 556–564, 2006.

[114] P. Hage and F. Harary, "Eccentricity and centrality in networks," *Social Networks*, vol. 17, no. 1, pp. 57–63, 1995.

[115] K. Stephenson and M. Zelen, "Rethinking centrality: Methods and examples," *Social Networks*, vol. 11, no. 1, pp. 1–37, 1989.

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

[117] S. Nanda and D. Kotz, "Localized bridging centrality for distributed network analysis," in *Proc. International Conference on Computer Communications and Networks*, St. Thomas, U.S. Virgin Islands, 2008, pp. 1–6.

[118] A.-M. Kermarrec, E. L. Merrer, B. Sericola, and G. Trédan, "Second order centrality: Distributed assessment of nodes criticity in complex networks," *Computer Communications*, vol. 34, no. 5, pp. 619–628, 2011.

[119] K. Wehmuth and A. Ziviani, "DACCER: Distributed assessment of the closeness centrality ranking in complex networks," *Computer Networks*, vol. 57, no. 13, pp. 2536–2548, 2013.

[120] W. Wang and C. Y. Tang, "Distributed estimation of closeness centrality," in *Proc. IEEE Conference on Decision and Control*, Osaka, Japan, 2015, pp. 4860–4865.

[121] ——, "Distributed computation of classic and exponential closeness on tree graphs," in *Proc. American Control Conference*, Portland, OR, 2014, pp. 2090–2095.

[122] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proceedings of the national academy of sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.

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

[124] S. Fortunato, "Community detection in graphs," *Physics reports*, vol. 486, no. 3, pp. 75–174, 2010.

[125] M. E. J. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Physical review E*, vol. 74, no. 3, p. 036104, 2006.

[126] B. H. Good, Y. A. de Montjoye, and A. Clauset, "Performance of modularity maximization in practical contexts," *Physical Review E*, vol. 81, no. 4, p. 046106, 2010.

[127] W. W. Zachary, "An information flow model for conflict and fission in small groups," *Journal of anthropological research*, pp. 452–473, 1977.

[128] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson, "The bottlenose dolphin community of Doubtful Sound features a large proportion of long-lasting associations," *Behavioral Ecology and Sociobiology*, vol. 54, no. 4, pp. 396–405, 2003.

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

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